IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 21 November 2024, accepted 16 December 2024, date of publication 26 December 2024,
date of current version 2 January 2025.

Digital Object Identifier 10.1109/ACCESS.2024.3523066

RESEARCH ARTICLE

Engineering End-to-End Remote Labs Using IoT-Based Retrofitting

SAVITHA VISWANADH KANDALA, AKSHIT GUREJA, NAGESH WALCHATWAR, RISHABH AGRAWAL, SHIVEN SINHA, SACHIN CHAUDHARI, (Senior Member, IEEE), KARTHIK VAIDHYANATHAN, VENKATESH CHOPPELLA, PRABHAKAR BHIMALAPURAM, HARIKUMAR KANDATH, (Member, IEEE), AND AFTAB HUSSAIN, (Member, IEEE)
International Institute of Information Technology Hyderabad (IIIT-H), Hyderabad, Telangana 500032, India

Corresponding author: Sachin Chaudhari (sachin.chaudhari@iiit.ac.in)

This work was supported in part by the Enabler Grants made available through the Kohli Center for Intelligent Systems (KCIS), International Institute of Information Technology Hyderabad (IIIT Hyderabad), Raj Reddy Center for Technology and Society (RCTS), and Technology Innovation Hub (TIH) Foundation for Internet of Things (IoT) & Internet of Everything (IoE), Indian Institute of Technology Bombay (IIT Bombay), under the CHANAKYA Fellowship Program (2022–2023) under Grant TIH-IoT/2023-03/HRD/CHANAKYA/SL/CFP-016.

ABSTRACT Remote labs are a groundbreaking development in the education industry, providing students with access to laboratory education anytime, anywhere. However, most remote labs are costly and difficult to scale, especially in developing countries. With this as a motivation, this paper proposes a new remote labs (RLabs) solution that includes two use case experiments: Vanishing Rod and Focal Length. The hardware experiments are built at a low-cost by retrofitting Internet of Things (IoT) components. They are also made portable by designing miniaturised and modular setups. The software architecture designed as part of the solution seamlessly supports the scalability of the experiments, offering compatibility with a wide range of hardware devices and IoT platforms. Additionally, it can live-stream remote experiments without needing dedicated server space for the stream. The software architecture also includes an automation suite that periodically checks the status of the experiments using computer vision (CV). The software architecture is further assessed for its latency and performance. RLabs is qualitatively evaluated against seven non-functional attributes - affordability, portability, scalability, compatibility, maintainability, usability, and universality. Finally, user feedback was collected from a group of students, and the scores indicate a positive response to the students' learning and the platform's usability.

INDEX TERMS Automated testing, Internet of Things (IoT), miniaturization, modular hardware design, peer-to-peer video streaming, remote labs, software platform.

I. INTRODUCTION

A. MOTIVATION

Several nations across the globe face issues in maintaining the quality of education due to factors like inadequate funding, a shortage of qualified teachers, and a lack of well-equipped laboratory facilities, particularly in rural areas [1], [2]. This situation is often exacerbated by external challenges such as conflicts, natural disasters, and health crises like the COVID-19 pandemic. The pandemic especially highlighted the vulnerability of the educational system worldwide, where global lockdowns and restrictions disrupted access to laboratory facilities, severely impacting science-based laboratory education [1], [3]. These constraints highlight the urgent need for innovative, flexible solutions to ensure uninterrupted, quality education under diverse circumstances. Remote lab is one such practical solution, which is the focus of this paper.

B. OVERVIEW OF REMOTE LABS

In remote labs, laboratory experiments situated at a given geographical location can be accessed from anywhere and

The associate editor coordinating the review of this manuscript and approving it for publication was Christian Pilato.

anytime in the world using the internet over a browser. Fig. 1 shows the overview of a generic IoT-based remote lab, which consists of two blocks connected by the internet- the remote block and the user block. The remote block consists of hardware at a remote location, which is usually developed and deployed within an institution or university. It includes the physical setup of the experiment, as well as sensors, actuators, cameras, and controllers. In such setups, cameras are used to stream the live feed of the experiment to the user. The user block consists of the internet platform at the user's side. The user can control the experiment parameters using the internet platform and view the live feed coming from the remote block. Experiment results can be presented in various formats on the internet platform, including tables, interactive plots and diagrams.

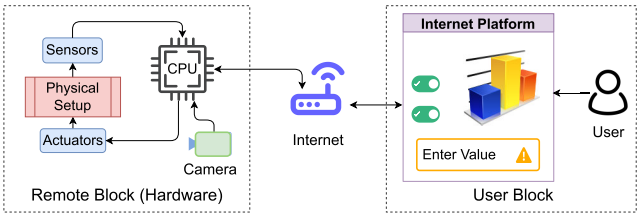


FIGURE 1. Overview of an IoT-based remote lab.

C. RELATED WORK

Numerous universities and research groups worldwide have implemented IoT-based remote lab solutions [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15]. Prominent examples include WebLab-Deusto and its spin-off, Labs-Land, both originating from the University of Deusto [6], [8], [16]. Their design of remote labs in different science and engineering domains comes with extensive features that primarily focus on building low-cost and reliable experiments by creating multiple instances of an experiment. UNILabs[1] [17], [18] is a network of web-based laboratories with several Spanish universities participating and hosting experiments that are shared over a Learning Management System [19]. Additionally, the Internet School Experimental System (iSES)[2] offers a wide range of experiments in physics and allied domains. The iSES Remote Lab SDK, comprising customisable JavaScript widgets and low-latency communication, enables programmers and non-programmers to create remote laboratory interfaces with data and video transfer facilities [7].

Specialised iterations of remote labs have emerged to cater to specific domains of science and engineering, including electronics and embedded systems [8], [9]. In [8], a multi-instance microcontroller-based embedded device remote laboratory is developed, surpassing state-of-the-art architectures in terms of scalability on low-cost setups. Furthermore, [9] introduces an open-source interface equipped with interchangeable circuit boards (plugs), providing educators with the flexibility to adapt circuits swiftly. Remote labs also teach programming languages like Python [10], [11], enabling users to program sensors and actuators, such as temperature sensors and LCDs. Several implementations of remote labs in the control engineering domain are discussed in [12], [13], and [14]. In [12], the authors introduce a low-cost, fully open-source remote laboratory for teaching automatic control systems. It includes complete instructions for building and replicating the remote laboratory's hardware and software components. In [13], a low-cost remote laboratory for control engineering experiments is developed using an inexpensive BeagleBone Black development board,[3] with experiments hosted on a dedicated Java application. In [14], the authors discuss the implementation of a remote lab for experiments hosted on restricted networks for control engineering education and evaluate the student experience using a survey. In [20], the authors have proposed a remote laboratory which uses YouTube for streaming video from the hardware to the users but lacks the real-time experience desired with such online remote experimentation experiences. Remote labs implementation in [21], proposes a 3-layer architecture and presents multi-user multiplexing to improve the scalability of remote labs but limits to electronic circuit-based experiments given such experiments provide near-instantaneous response times.

A considerable amount of work has also been done from the pedagogical aspect on the various methods of delivering remote labs. Multiple articles in the literature aim to increase user engagement in deployed remote labs through varying methods of delivery such as active learning [15], [22], flipped classrooms [23], gamification [24] and project-based learning [25]. Active learning is a teaching methodology which aims to encourage students to learn by actively participating in discussions and hands-on activities rather than just passively listening to lectures. A flipped classroom is a teaching method that is based on the principle of active learning, where the students read, learn and review the course material before the class and during class, engage in collaborative exercises, activities, and discussions. Gamification, on the other hand, is an educational approach that incorporates game principles and elements such as points, badges, challenges, and leaderboards to increase student engagement and learning. Project-based learning allows students to learn by working on real-world projects aiming for a better understanding through practical application. Studies also suggest the need for diversification of distance learning activities to stimulate student motivation [26].

D. ATTRIBUTES OF REMOTE LABS

As mentioned previously, the literature presents numerous remote lab solutions (systems), each with its features and capabilities. To establish the suitability of these systems, the non-functional attributes (NFAs), also referred to as non-functional requirements [27], are considered. There are

[1]https://unilabs.dia.uned.es/
[2]https://www.ises.info/index.php/en
[3]https://www.beagleboard.org/boards/beaglebone-black

several NFAs, and they define how a system will work by laying out its requirements and limitations. It is crucial to establish a common ground, and to this end, this paper has identified seven crucial NFAs and classified relevant features under each NFA. These NFAs and features serve as critical comparison criteria for remote laboratory solutions. The identified NFAs and their features are as follows:

1) **Affordability**: It describes the cost-effectiveness of a remote labs system, primarily focusing on *low-cost designs*. The cost factor is particularly crucial for making labs affordable and accessible to a large number of students in developing countries like India.
2) **Portability**: It describes the ease with which hardware experiments can be transported from one location to another. This can be achieved by designing *modular setups*.
3) **Scalability**: It describes the system's capacity to grow and adapt to increasing demands. This can be achieved in multiple ways:
   - *Adding hardware (HW) instances to the platform* - denotes the capability of adding multiple instances of the same experiment to the remote lab's platform.
   - *Miniaturised HW setups* - refers to designing remote experiments that are more compact than traditional laboratory experiments.
   - *Peer-to-Peer (P2P) live video streaming* - refers to a live video streaming architecture that does not require a centralised streaming server.
4) **Compatibility**: It describes the system's ability to integrate and operate with a diverse range of configurations. This includes several features of the system:
   - Compatibility with *different hardware boards* for controlling the actuators and sensors and connect to the internet for data transmission (like ESP32[4] and Raspberry Pi[5]).
   - Compatibility with *multiple IoT platforms* for receiving data from different hardware boards and making the data accessible online (like Blynk[6] and Thingspeak[7]).
   - Compatibility with *various types of cameras* for live video streaming of the experiments. Cameras can include standalone cameras like IP cameras and non-standalone cameras like USB cameras.
5) **Maintainability**: It describes the efforts required to keep the system running normally. *Automated testing* showcases the system's ability to automatically report any errors affecting its functionality.
6) **Usability**: It describes how user-friendly the system is and is best known from *user feedback* collected from individuals (users) who have used the system.
7) **Universality**: It refers to the system's ability to be accessed across a variety of devices (mobiles, desktops), operating systems (Windows, iOS, Android), and web browsers (Firefox, Edge, Chrome), maintaining a consistent and adaptable user experience.

These NFAs offer a comprehensive basis for evaluating and qualitatively comparing various remote lab solutions, providing valuable insights into different solutions' strengths and weaknesses. Recent works ($\leq$ eight years) that provide proper implementation details of the hardware and software components have been considered to evaluate prominent and relevant remote lab solutions in the literature. Since different works have distinct remote lab architectures, some of which are quite resource-extensive and implement various types of experiments, it is challenging to replicate and compare them quantitatively. Thus, the paper provides a comparison of existing and proposed remote laboratories based on qualitative features. Moreover, The focus is on labs operating within web browsers without external plugins rather than dedicated applications. According to the above criteria, the chosen remote lab implementations are: LabsLand [6], [8], [16], RaspyLab [11], RaspyControl Lab [12] and iSES [7]. Table 1 compares the chosen remote labs among the identified NFAs and the proposed solution in this paper.

E. CONTRIBUTIONS OF THE PAPER

The contributions of this paper are as follows:

- An end-to-end remote labs solution, known as **RLabs** (short for **R**emote **Labs**), has been implemented and deployed on the campus of the International Institute of Information Technology (IIIT) in Hyderabad, India.
- Two use case experiments, Vanishing Rods and Focal Length, based on high-school physics, are designed by retrofitting IoT components to the traditional laboratory equipment.[8]
- Additionally, modular and miniaturised versions of the two use case experiments are also proposed and built[9] to make the setups portable and scalable.
- A novel software architecture for RLabs has been proposed and implemented. This architecture offers several novel features that facilitate the seamless scalability of remote experiments. This includes the introduction of an interoperability layer to support multiple IoT platforms, and the implementation of P2P live video streaming.
- An automatic testing system is designed to verify if the remote experiments are functioning as intended, eliminating the need for manual checks. This system includes the usage of computer-vision (CV) algorithms

[4]https://www.espressif.com/en/products/socs/esp32
[5]https://www.raspberrypi.com/products/
[6]https://blynk.io/
[7]https://thingspeak.com/

[8]These experiments are chosen to showcase the capabilities of the built system. Additional experiments from various domains and existing remote laboratories can also be selected and integrated independent of the hardware's complexity as long as it can be connected to a microprocessor or microcontroller board.

[9]Initial results on miniaturization are presented in [20]. This work introduces a new concept of portability through modularity, along with other unique features, making it different from previous works.

TABLE 1. Feature comparison of various remote labs.

Attribute	Features	LabsLand [6], [8], [16]	RaspyLab [11]	RaspyControl [12]	iSES [7]	**RLabs** (Proposed)
Affordability	Low-cost setups	✓	✓	✓	×	✓
Portability	Modular hardware setups	×	×	×	×	✓
Scalability	Adding HW instances to the platform	✓	✓	×	✓	✓
	Miniaturised HW setups	×	×	×	×	✓
	Peer-to-Peer live video streaming	×	×	×	×	✓
Compatibility	Tested different HW boards	✓	×	×	✓	✓
	Tested multiple IoT platforms	×	×	×	×	✓
	Tested different cameras	✓	×	×	×	✓
Maintainability	Automated testing	✓	×	×	×	✓
Usability	User feedback	✓	✓	×	✓	✓
Universality	Device-friendly	✓	✓	✓	✓	✓

✓Implemented × Not Implemented

and Selenium-based automation and is integrated with the proposed RLabs software architecture.
- Latency and server performance has been evaluated for the RLabs software architecture as more users perform remote experimentation.
- User feedback from forty-five students located at a different geographical city is collected and reviewed, containing questions to rate their usability and learning outcomes after using the built RLabs system.

The proposed work is novel concerning the relevant recent remote lab implementations as shown in Table 1. The existing implementations lack features like modular and miniaturised hardware experimental setups, and P2P live video streaming and did not test different IoT platforms, leading to portability, scalability and compatibility limitations. All the implementations mentioned in Table 1 have institute-retained hardware only. RaspyControl Lab in [12] also provides instructions for other universities, organizations, and students to replicate the remote labs with their own hardware. The proposed RLabs solution is institute-retained, but by virtue of its portability, affordability, and universality, it can also be used to host the experiments developed by students or other universities themselves. Additionally, the systems in [7], [11], and [12] have not tested different cameras for live-streaming and lack an automation testing suite that limits the compatibility and maintainability of the systems. In [7], the setups are not low-cost as they use personal computers for connecting the experiments to the internet, affecting the affordability of the systems. In [12], different hardware instances of an experiment cannot be added, and a user feedback is not presented that impacts the scalability and questions the system's usability. In [11], different hardware boards are not tested to design the experiments that limit the system's scalability. The proposed work provides a remote lab implementation which can be integrated into pedagogical frameworks such as flipped classrooms, blended, active learning and more to increase student engagement.

F. STRUCTURE OF THE PAPER

The structure of the paper is outlined as follows. In Section II, the theory of the two use case experiments is briefly presented. Section III provides detailed insights into the hardware components used for retrofitting the experiments and creating their miniaturised versions. Section IV elaborates on the software platform designed for hosting and managing these experiments. It also discusses the automated testing suite and presents the evaluation of the software architecture for its latency and performance. In Section V, results are articulated systematically by analysing the performance and effectiveness of our model against each proposed attribute. Finally, Section VI concludes the work presented and presents the scope for future work.

II. USE CASE EXPERIMENTS

In this section, two considered use case experiments based on fundamental high-school physics concepts are explained in detail, including their aim, theory, methodology, and results. As mentioned earlier, the experiments considered below are selected to demonstrate the functionality of the RLabs system. Experiments involving more complex hardware, such as titration [28], can be easily added to the proposed RLabs system.

A. EXPERIMENT 1: THE VANISHING ROD EXPERIMENT

1) AIM

To observe the change in visibility of a glass rod when immersed in oil and in water media

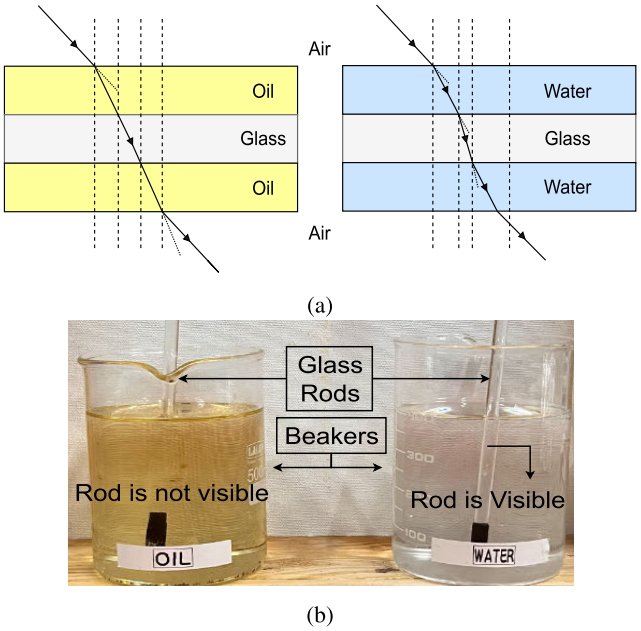


FIGURE 2. (a) Ray diagram of light entering different surfaces. (b) Visibility of glass rods when dipped in beakers containing water and oil separately.

2) THEORY

The concept of refractive index is explored in this experiment. While light travels from one medium (e.g. air) to another (e.g. water), the bending of light can be observed. This happens due to the slowing down of the speed of light when the medium changes. Every material is associated with a refractive index that quantifies the refraction of light. The higher the refractive index, the higher the deviation of light in the entering medium. However, suppose another medium with a numerically closer refractive index surrounds an object. Then, the object appears to have disappeared in the medium, as there will be no reflection and refraction of the light while passing through the object, as illustrated in Fig. 2.

3) METHODOLOGY

Fig. 2b shows the apparatus used in the setup which includes borosilicate glass rods (Refractive index $\mu = 1.5$), sunflower oil ($\mu = 1.47$), drinking water ($\mu = 1.36$), borosilicate glass beakers ($\mu = 1.5$). One beaker is filled with sunflower oil and the other beaker is filled with water. The experiment is performed by placing the glass rods into the beakers and observing the visibility of glass rods in the beakers.

4) RESULTS

Fig. 2b depicts the visibility of glass rods when immersed in glass beakers. It can be observed that the glass rod dipped in the oil beaker tends to disappear while the other glass rod remains clearly visible. This is attributed to the fact that the glass rod and sunflower oil have very similar refractive index values that make the glass rod vanish in the oil medium. However, the glass rod and water have different refractive index values that make the glass rod visible in the water medium.

B. EXPERIMENT 2: FOCAL LENGTH EXPERIMENT

1) AIM

To determine the focal length of a biconvex lens by forming a sharp image of an object on the screen.

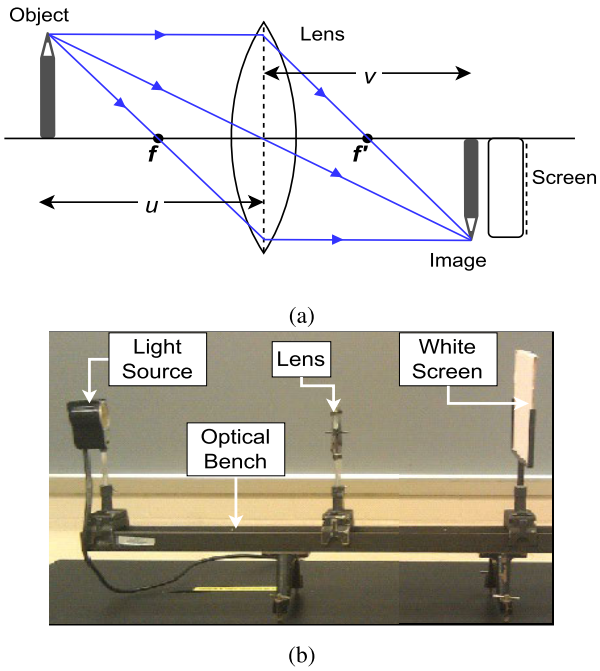


FIGURE 3. (a) Ray diagram of a thin biconvex lens. (b) Experimental setup of the Focal Length experiment (source: hirophysics.com).

2) THEORY

The experiment focuses on the field of optics and aims to determine the focal length of a biconvex lens [29]. The focal length is a measure that defines the distance between the lens and the point where light rays converge to form a sharp image. It determines the lens's viewing angle and the magnification of the image produced. To determine the focal length of a given thin biconvex lens, an experiment is set to get a sharp image of an object on a screen by adjusting the distances of the screen and object from the pole of the lens as shown in Fig. 3a. Once a sharp image is formed on the screen, the object distance (u) and the image distance (v) are used to calculate the focal length (f) of the used lens using the lens formula with proper sign conventions. Any measurement towards the direction of the incident ray is considered positive, and distances opposite to the direction of the incident ray are negative. In the case of a thin biconvex lens, as shown in Fig. 3a, the object is on the left side of the lens and the real image will form on the right side if the object distance is greater than f. So according to the sign convention, u will always be negative and v will always be positive for real

images. Then the modified lens formula would be $\frac{1}{f} = \frac{1}{v} + \frac{1}{u}$, where u, v and f would be the absolute values.

3) METHODOLOGY

Fig. 3b shows the apparatus used in the setup which includes a white screen, a light source, a biconvex glass lens and an optical bench upon which the experiment is performed. Firstly, the biconvex glass lens is placed on the lens stand. The light source and the white screen are placed on the either sides of the lens such that the object, lens and the screen lie on a straight line. The experiment is performed by adjusting the positions of both the object and the screen platforms and observing the image formation on the screen.

4) RESULTS

Table 2 displays a few pairs of u and v values that yield a sharp image during the experimentation. It is notable that multiple sets of values can result in a clear and sharp image. Irrespective of these different values observed, the lens's focal length remains the same, given that the lens used is the same in every trial.

TABLE 2. Focal Length calculations for different values of u,v for which sharp images are obtained.

S.No	u (cm)	v (cm)	f (cm)	% error
1	20.59	20.38	10.24	2.4
2	29.5	15.89	10.33	3.3
3	42.65	13.95	10.51	5.1

TABLE 3. Comparison of different stepper motors used in the experiments.

Motor Type	28BYJ-48	Nema 17
Step Angle	0.0875°	1.8°
Rated Voltage (V)	5	12-48
Rated Current (A)	0.4	1.68
Holding Torque (kg-cm)	0.34	4.2
Step Accuracy	Low	High
Size (mm^3)	34×18×10	40×42×42
Weight (g)	∼50	∼300

III. HARDWARE: DESIGN AND IMPLEMENTATION

In this section, hardware designs of retrofitted lab-scale remote experiments are presented along with their design choices. Later, more compact lab-scale versions are also presented, emphasising their modular and miniature designs.

A. RETROFITTED LAB-SCALE EXPERIMENTAL SETUPS

The lab-scale setups are created by retrofitting IoT components into the existing experimental equipment used in traditional laboratories, offering users a visual experience similar to traditional lab experiments.

1) EXPERIMENT 1 - VANISHING ROD

The apparatus required for the traditional experiment is already discussed in Section II. Additional apparatus required to retrofit the experiment for remote accessibility includes plywood, 28BYJ-48 stepper motors (Table 3), Raspberry Pi 3B+ and a Raspberry Pi camera (RaspiCam). Figs. 4a, 4b, and 4c show the physical setup and circuit design of the Vanishing Rod experiment. One beaker is filled with sunflower oil, and the other is filled with water. They are then placed near the base, which is prepared from plywood. The glass rods are attached with strings to the stepper motors fixed above the beakers. The RaspiCam is positioned to capture both beakers within its field of view in the video feed. This feed is sent to Raspberry Pi 3B+, which controls the motors. The experiment is performed by moving the glass rods into the beakers placed and observing the visibility of the glass rods in the beakers. Both the glass rods are moved up or down vertically from the beakers simultaneously by the stepper motors connected to the Raspberry Pi. Users can control the glass rods' movement, and the glass rods can either be dipped into the beakers or not.

The Raspberry Pi 3B+[10] is chosen as the primary board for several reasons. Firstly, it boasts multiple GPIO pins, enabling interaction with numerous sensors and actuators simultaneously. Secondly, its compatibility with a Linux operating system facilitates programming in various languages. Thirdly, it supports various cameras like the RaspiCam and the USB cameras. Specifically, the RaspiCam, which is used for live-streaming the experiment, can be connected via the Camera Serial Interface (CSI) to the Raspberry Pi. The CSI is advantageous due to its high data throughput, ensuring real-time image and video processing [30]. Lastly, the Raspberry Pi 3B+ provides versatile internet connectivity options via Wi-Fi or an Ethernet port. Different types of motors can be used for the actuation, such as servo motors and stepper motors [31]. Each motor serves a specific purpose, varying in accuracy, cost, and delivered torque. In this work, stepper motors are primarily utilised due to their cost-effectiveness, aligning with the requirements of the presented experiments.

2) EXPERIMENT 2 - FOCAL LENGTH

The apparatus required for the traditional experiment is already discussed in Section II. Additional apparatus required to retrofit the experiment for remote accessibility includes two NEMA 17 stepper motors (Table 3), two A4988 micro-stepping drivers, two limit switches (end-stop switches), a light source, a Raspberry Pi 3B+, a RaspiCam and an Ant Esports USB camera.[11] Figs. 5 and 6 show the experiment's physical setup and circuit design. Stepper motors are used to move the object and screen platforms horizontally while the position of the lens is fixed. The lens's focal length is calculated by considering the distances moved by the screen and object platforms from the lens once a sharp image is formed.

[10] https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/
[11] antesports.com/product/ant-esports-streamcam120-1080p-hd-webcam/

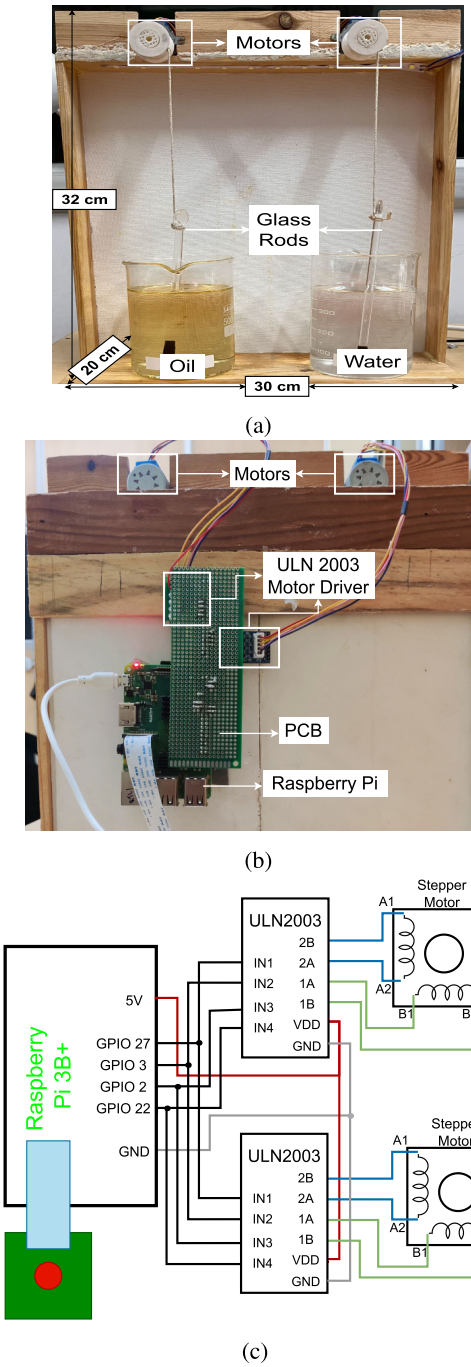


FIGURE 4. Hardware description of the lab-scale Vanishing Rod experiment. (a) Front-view of the experimental setup. (b) Back-view of the experimental setup. (c) Circuit diagram of the setup.

The stepper motors used for this experiment differ from those used in the Vanishing Rod experiment as more accuracy and torque are required (Table 3). The motors are attached to a screw shaft on which the screen/object is mounted, which converts the rotational motion to linear motion. Individual sliders facilitate the movement of the object and screen independently. The distance moved by the object and screen is linearly proportional to the number of steps/ degrees rotated by the stepper motor, which moves in precise and repeatable increments, which allows for consistent and accurate movement of the object and the screen. Micro-stepping drivers (A4988) control the motors, which improve the movement of the screen and object platforms. This micro-stepping driver allows the motor to take 400 steps per revolution. Here, 1 step is equivalent to 10 µm and all the movements are based on this relation.

Limit switches are placed to recalibrate the motors that control the object and screen movements. The light source, which is electrically powered, is used as an object to observe the inversion in the image formed by the lens. The full-sized optical bench allows users to explore different image formations through the biconvex lens. RaspiCam is positioned to capture the white screen on which the image is formed, while the USB camera captures the side-view of the entire experimental setup. A USB camera is used instead of another RaspiCam as a single Raspberry Pi 3B+ can only handle one RaspiCam interfaced using CSI. The captured feed from both cameras is sent to the Raspberry Pi, which also controls the motors and receives signals from the limit switches.

B. MINIATURISED EXPERIMENTAL SETUPS

Miniaturised setups are smaller replicas of the lab-scale setups, primarily constructed using 3D printed components and commonly available materials that require no welding, adhesive bonding, or specialised mechanical tools typically used to build lab-scale setups. The designs are intentionally crafted for easy assembly, allowing users to follow a manual for straightforward construction. Minimal tools, such as a screwdriver and soldering iron, are sufficient for assembly.

1) EXPERIMENT 1: VANISHING ROD

Figs. 7 show the miniaturised Vanishing Rod experiment. The miniaturised setup primarily consists of an ESP32, Raspberry Pi Zero 2 W, a single 28BYJ-48 stepper motor, ULN2003 motor driver, two pulleys, two borosilicate glass rods, and two 50mL beakers filled with sunflower oil and water. The exoskeleton of the experiment is entirely 3D printed in multiple smaller parts that can be assembled like puzzle pieces within a few minutes. ESP32 board controls the hardware components that save space without affecting any functionality of the experiment. Slots are provided on the exoskeleton to fit the beakers, glass rods and electronic components that are part of a single PCB, as shown in Fig. 7c. The glass rods are tied to a small rectangular platform (rod holder) spooled over a pulley connected to a single motor that rotates clockwise and anticlockwise. This platform is designed to move in a linear path — up and down. It is constrained into a slit which blocks it from rotating and toppling, keeping rods stable when moving, whereas, in the lab-scale experimental setup, the rods can freely rotate and swing when rods are moving. The slit which constrains this platform is indicated by the arrow in Fig. 7b.

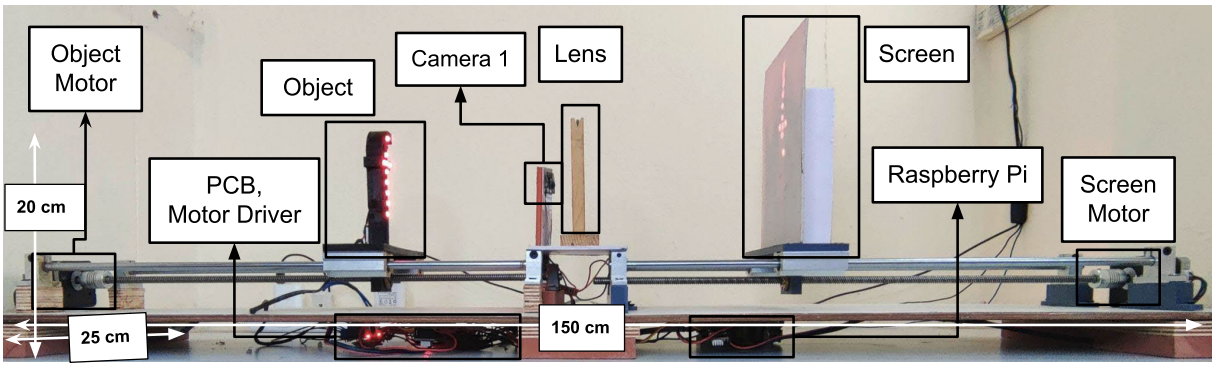


FIGURE 5. Side-view of the experimental setup for focal length experiment.

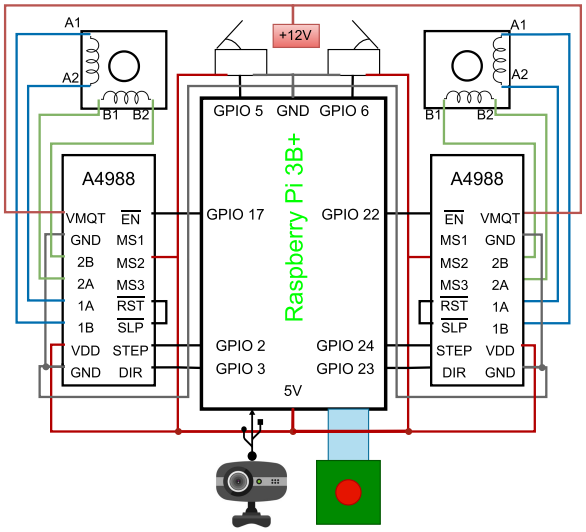


FIGURE 6. Circuit diagram of the focal length setup.

Raspberry Pi Zero 2 W is a more compact version of the Raspberry Pi 3B+ with a CSI port for video streaming and Wi-Fi connectivity. However, it has slightly lower processing capabilities and lacks an Ethernet port. It is important to note that the streaming pipeline is decoupled from the experiment controls in specific setups to showcase the compatibility of various hardware boards and cameras. This is discussed in detail in Section V.

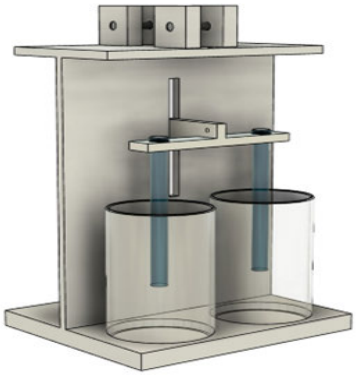
2) EXPERIMENT 2: FOCAL LENGTH

Fig. 8 shows the miniaturised Focal Length experiment. The miniaturised setup apparatus employs a design similar to that of 3D printers, comprising two A4988 drivers, NEMA 17 stepper motors, limit switches, two Raspberry Pi Zero 2 W[12] and two RaspiCams. The body is constructed from V-slot aluminium profiles, which serve as a versatile base for attaching various components. Two such profiles are used, one each for the object and screen. Using these aluminium profiles as a base, the rest of the structure is built, which includes the mechanism for moving the platform upon which the object/ screen is placed, space for motor operation and slots for the electronics. The other structural components, like the screen and object platforms and supports for the motors, are 3D-printed. These 3D-printed parts are designed to have slots and holes for screws appropriately, which can be fixed to the aluminium extrusion profiles using sliding nuts (slide and lock mechanism), eliminating the need for drilling holes in the profiles. In contrast to the traditional linear screw actuation, this setup utilises a belt drive mechanism to manoeuvre the object and screen platforms. This modification reduces the number of components and decreases the overall weight of the apparatus. A timing belt facilitates the movement of the screen, and object looped over a pulley and the stepper motor. Once the two profiles are built with the motors joined at the ends, the profiles are joined with a joint 3D-printed base upon which the lens stand is mounted. This 3D-printed base has connections between the motors and the illuminated object. These connections are made using JST and DuPont connector wires, which enable plug-and-play links to the electronics box containing the experiment's circuitry, as depicted in Fig. 8. The PCB inside the electronics box provides slots for connecting the Raspberry Pi, buck converter, and motor drivers. Additionally, a power supply not included in the setup can be directly connected through the power socket on the box's side. A Raspberry Pi and a RaspiCam are placed to stream the side-view of the experiment. A single Raspberry Pi could not handle the streaming from both a RaspiCam and a USB camera; hence, two Raspberry Pis were used.

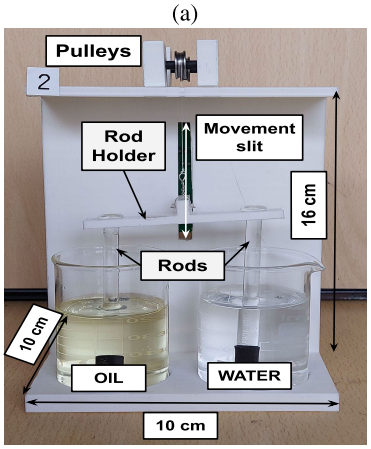
[12]https://www.raspberrypi.com/products/raspberry-pi-zero-2-w/

IV. SOFTWARE: RLABS PLATFORM, IMPLEMENTATION AND TESTING

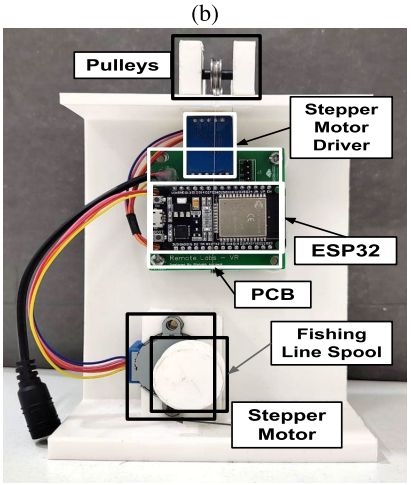
For the hardware devices described earlier, there is need for a software platform for user accessibility and a powerful and efficient dashboard for proper data management. As depicted in Fig. 1, the user interaction is enabled for such platforms via the internet platform. Remote lab solutions, including RLabs, are built on the foundational backbone that anyone can operate the hardware setups remotely. The internet platform



(a)



(b)



(c)

FIGURE 7. Hardware description of the miniaturised vanishing rod setup. (a) 3D model of the setup. (b) Front-view of the setup. (c) Back-view of the setup.

developed as part of RLabs has been referred to as RLabs platform henceforth. The RLab platform and devices must be accessible to anyone, anywhere and anytime [7]. For this purpose, the software platform must function appropriately on all smartphones, tablets, laptops and PCs without requiring much of additional applications [32] for increased user accessibility. Moreover, for a platform enabling remote experimentation, ensuring the reliability of an experiment's outputs is crucial, as inconsistencies can impact the validity of the results obtained. This is particularly challenging in experiments that involve moving mechanical components due to the inherent complexities and potential for errors. In light of this, the browser-based RLabs platform that facilitates the conduction of experiments remotely by the users, is complemented by a comprehensive automated testing system. The developed automated testing system is invoked periodically to emulate a virtual user and ensure the proper working of the platform as a whole. It is crucial to note that none of the known state-of-the-art remote labs have a CV-aided automated testing system, and this is thus another major novelty of our proposed architecture. Fig. 9 provides a high-level overview of the end-to-end RLabs software solution, and a detailed description of the implementation is discussed in the rest of the section.

A. RLABS PLATFORM

The proposed RLabs platform in Fig. 10 presents four major components required for remote experimentation - *the experiment frontend, the backend, the interoperability cloud service, along with the IoT component*. As the primary aim of RLabs is to conduct scientific experiments remotely based on user inputs, RLabs platform must have an interactive frontend for taking user inputs and a robust backend to relay these parameters and handle client requests. The communication between the frontend, backend and the experiments is a crucial aspect that must be considered while solutions requiring low-cost and low-latency are being built. An interoperability cloud service to facilitate the data exchange between multiple components and hardware devices using IoT for remote experimentation. Fig. 10 uses standard UML (Unified Modeling Language) syntax to present an overview of these components and the interface between them. A standard way that the backend servers and software platforms, in general, employ to establish communication to the IoT component is by using HTTP(S) GET and POST requests to the interoperability layer. Moreover, HTTP(S)-based REST APIs provided by IoT platforms such as Blynk[13] can be configured to work uniquely to our requirements and has been presented as the middleware and interoperability service in Fig. 10. It is important to mention that Blynk Cloud forming the interoperability component, can be substituted with any alternative if it provides the service of relaying the information between the backend and IoT component. The IoT component built for RLabs constitutes the hardware like Raspberry Pi, sensors and actuators, which take the user inputs relayed by the interoperability component. Chromium browser in the Raspberry Pi is used to provide the WebSocket interface to the backend component.

RLabs software platform allows users to connect and control hardware remotely for conducting science experiments.

[13]https://blynk.io/

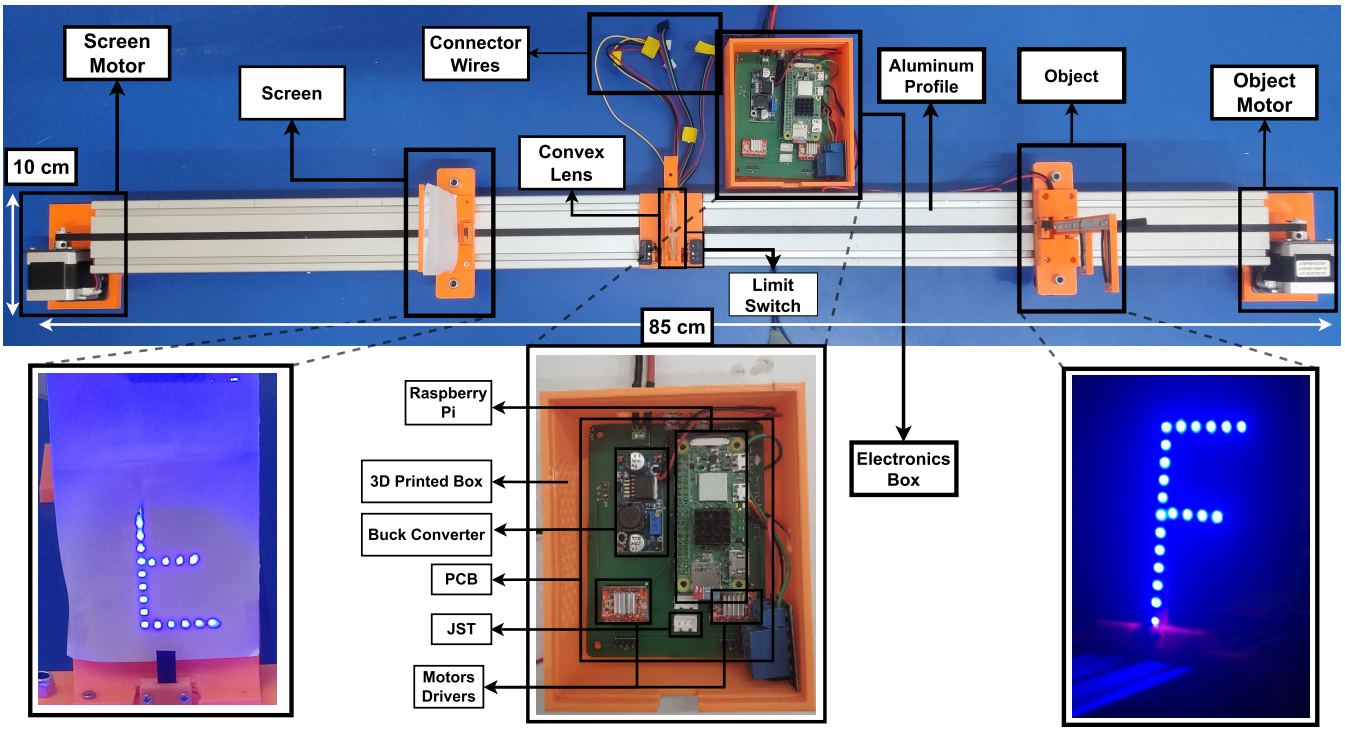


FIGURE 8. Hardware description of the miniaturised focal length setup.

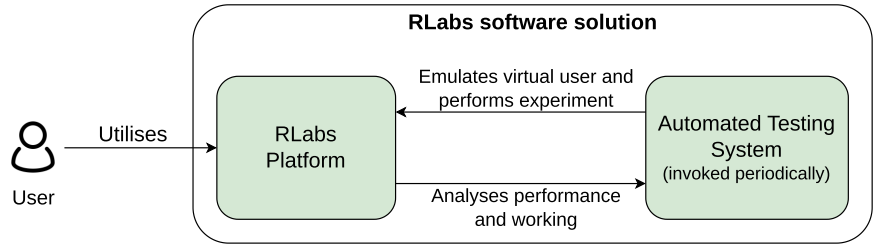


FIGURE 9. Overview of RLabs software solution.

Such platforms require the user to be able to view the outputs and results in real-time. This makes low-latency video streaming an essential feature of RLabs, especially when the platform involves the mechanical movement of hardware using actuators. As depicted in Fig. 10, the RLabs platform makes use of an open-source implementation of the WebRTC protocol to enable P2P communication and live stream the video feed of the RaspiCam captured by the Chromium browser directly to the user's frontend. Hence, user interaction with the experiments is possible in RLabs because of the perfect sync between the components and their integrated working creating a workflow unique to the framework, which has been explained in detail below.

B. EXECUTION FLOW

Fig. 11 shows the sequence diagram for the RLabs web-based platform following standard UML syntax with an appropriate legend to depict the communication protocol used between the components mentioned above for an unoccupied experiment hardware node. It provides a visual depiction of the execution flow and communication between the hardware and software components required to coordinate and relay information intricately to perform remote experimentation successfully using the holistic RLabs system. Any user experimentation comprises three initiation steps as depicted in Fig.11- the user requesting experiment access, the user providing input parameters and the user exiting the experiment, each of which starts a unique communication flow. However, for this user-input and hardware-output-based system, the IoT embedded system first needs to inform the central backend server that it is active for remote experimentation which is performed by the Raspberry Pi of the experiment by sending its private credentials and experiment ID. An experiment ID is provided to each Raspberry Pi, to recognize which experiment a particular Raspberry Pi corresponds to. Moreover, some hardware

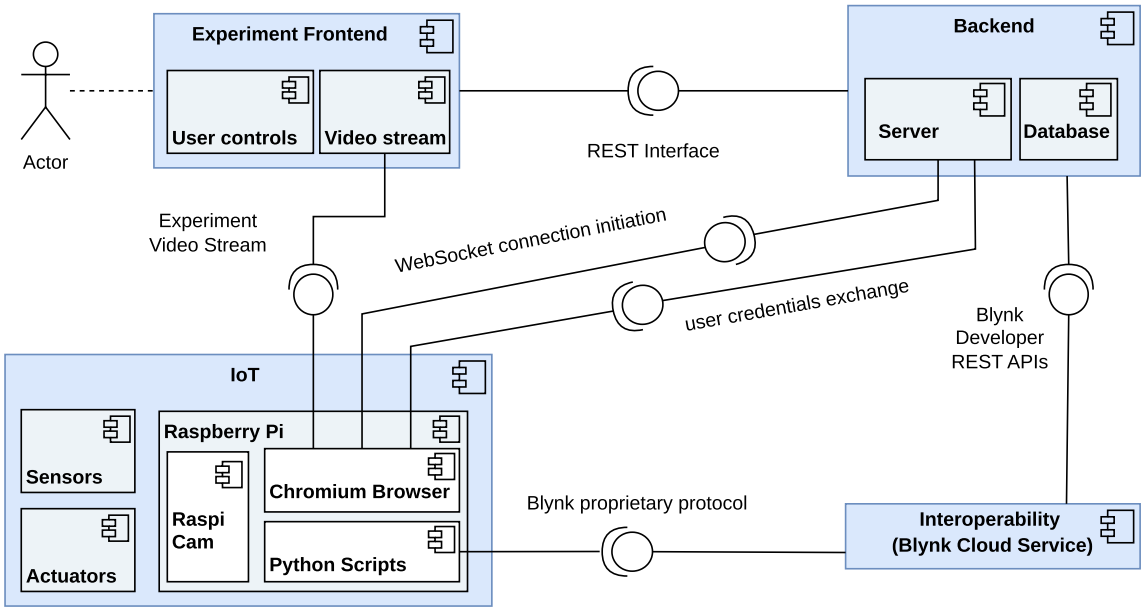


FIGURE 10. Major components of RLabs platform.

nodes for experiments might have two or more Raspberry Pis sending the output value and video stream, due to which each Raspberry Pi requires its unique private credentials. By sending these two values, the Raspberry Pi is authorised by the central backend server, with the experiment hardware node now ready to take in user inputs, conduct experiments and generate output.

Fig 12 shows the user's perspective while accessing a particular experiment[14] hosted on the RLabs platform To perform an experiment, a user first must sign up and log in for verification and authorization on the RLabs platform.[15] Once logged in, the user on the homepage can choose to access any experiment. As shown in Fig. 11, the first initiation step is where the user chooses the desired experiment to perform, after which the user is verified and authenticated again to prevent invalid access, and the PeerID generated automatically by the browser is sent to the backend server. Upon receiving the user PeerID successfully, a triple-fold check is performed for a hardware node's availability. A flag on the Blynk cloud is maintained for each experiment, which checks whether an experiment is under use. The connection of the hardware node with the Blynk platform for data exchange, flag value and the presence of a working RaspiCam stream are checked before marking an experiment as available and unoccupied for the current user. To check if the RaspiCam video stream is working, the user PeerID is sent to the Raspberry Pi, which responds with an acknowledgement if the service is active, and a WebRTC call with the video stream of the experiment is made by the Raspberry Pi directly to the user's frontend by connecting to the sent user PeerID. Once the experiment video stream from the Raspberry Pi has commenced, both frontend and IoT components are active and ready for user inputs. However, if an acknowledgement is not received in under five seconds, the experiment is marked unavailable and currently offline for experimentation. As shown in Fig. 12, once the user crosses all verifications post experiment selection, and the triple-fold check for a node's availability is completed and passed, the user is now redirected to the dedicated webpage for the experiment with the screen divided into three major sections - i) the instruction panel on the left with the experiment's theory and guide to conducting the experiment, ii) the real-time stream of the hardware setup in the centre and iii) the control panel which includes buttons, sliders and more to allow the users to interact with the experiment's hardware node. There is also a dedicated timer on the top-right of the screen depicting the time remaining with the user to experiment.

The frontend client provides a user interface for users to interact with the hardware experiments. The second initiation step is when the frontend client provides inputs to a particular experiment using interactive buttons, sliders and switches in the control panel, the server-client connection is established, and the information payload consisting of the input values is sent to the backend server. These are then communicated to Raspberry Pis associated with the hardware setups using the Blynk IoT platform. The experiment is conducted using the user-input parameters, and the output recorded is updated by the Raspberry Pis back on the Blynk platform. The server retrieves these outputs using GET API calls made to the Blynk Cloud, which are then communicated to the frontend, where they are displayed using graphs, charts, and tables as part of the graphical user interface (GUI).

[14]Demo video: https://remote-labs.in/demo
[15]https://remote-labs.in

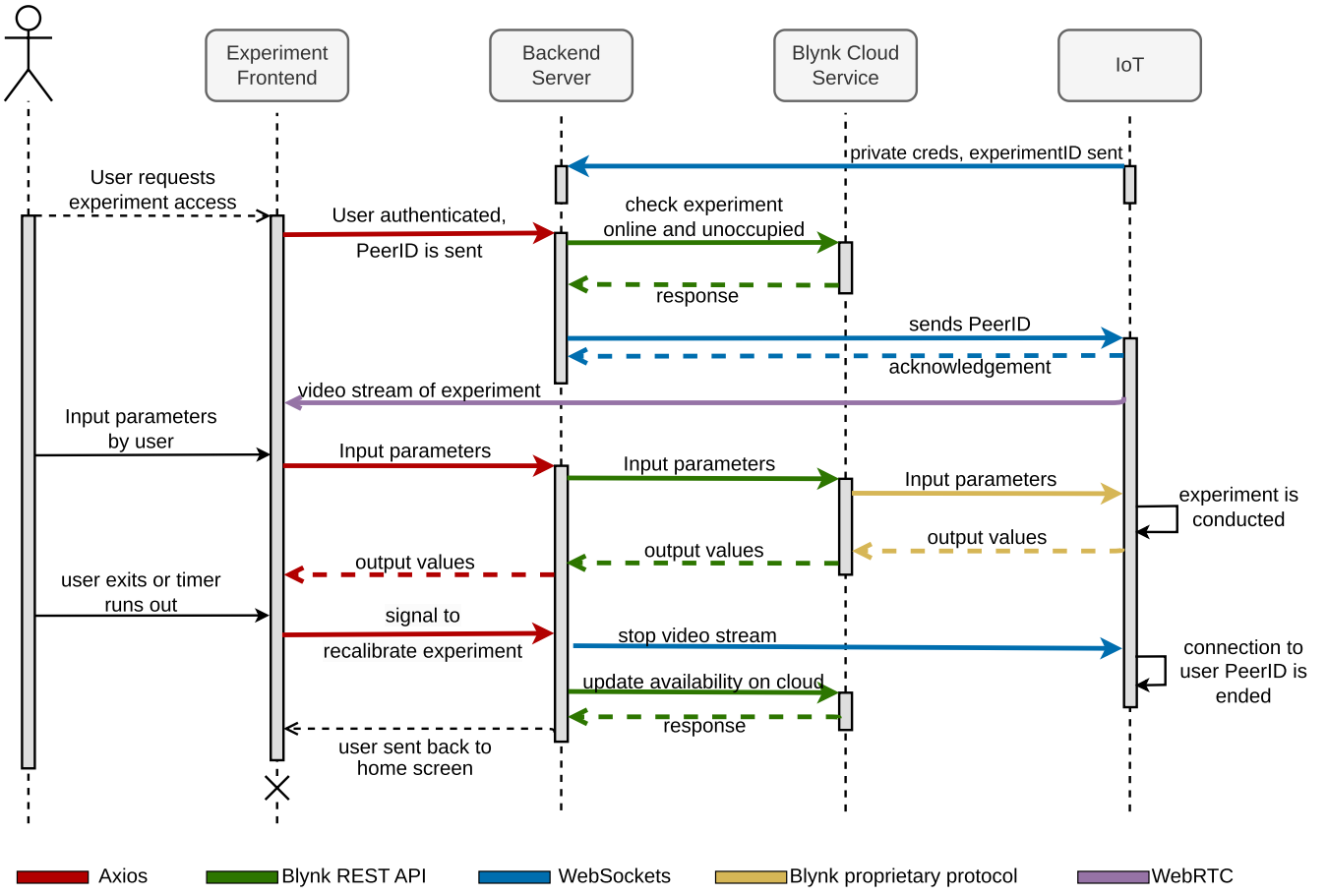


FIGURE 11. Sequence diagram for remote experimentation using the RLabs platform.

Finally, when a user leaves, or the session times out, the experiment needs to recalibrate and be marked unoccupied so that it's available for the next user. For this purpose, a signal is sent to the Blynk Cloud to recalibrate the hardware experiment and change the flag value to unoccupied. The Raspberry Pi is also signalled to disconnect with the current user PeerID. As depicted in Fig. 12, the user is then redirected to the home page while the next client in the queue is permitted to enter this experiment and the flow repeats for each user remote experimentation. The components and features integral to the platform's implementation and enabling the student to use the interface are discussed in detail below.

C. KEY ARCHITECTURAL DESIGN DECISIONS

Fig. 13 presents the technologies used in implementing the four major components constituting RLabs, their subcomponents along with the protocols enabling communication between them and are explained in detail below.

1) EXPERIMENT FRONTEND

The experiment frontend enables user interaction, allowing the client to provide input parameters through the buttons, sliders, and switches that form the user interface. Moreover, to view real-time outputs of the experiment, the users also require the experiment's video stream, which is crucial to any frontend enabling remote experimentation. The technologies used to implement the subcomponents - user interface and P2P video streaming in RLabs are provided in Fig. 13 and are explained below.

a: USER INTERFACE

The RLabs Document Object Model (DOM) and frontend are implemented using HTML5, CSS3 and JavaScript. HTML5 is a highly standardised language that helps provide structure to web pages and allows browsers to understand the content so that it can be displayed according to the styling and layout defined using the CSS3 language. Tailwind[16] and Material UI[17] are styling libraries which speed up the development process by providing pre-built CSS classes for styling, offering the opportunity to implement customisable user interfaces. JavaScript allows for running event-driven logic in the student's browser [33]. React,[18] which is a popular

[16]https://tailwindcss.com/
[17]https://mui.com/material-ui/
[18]https://react.dev/

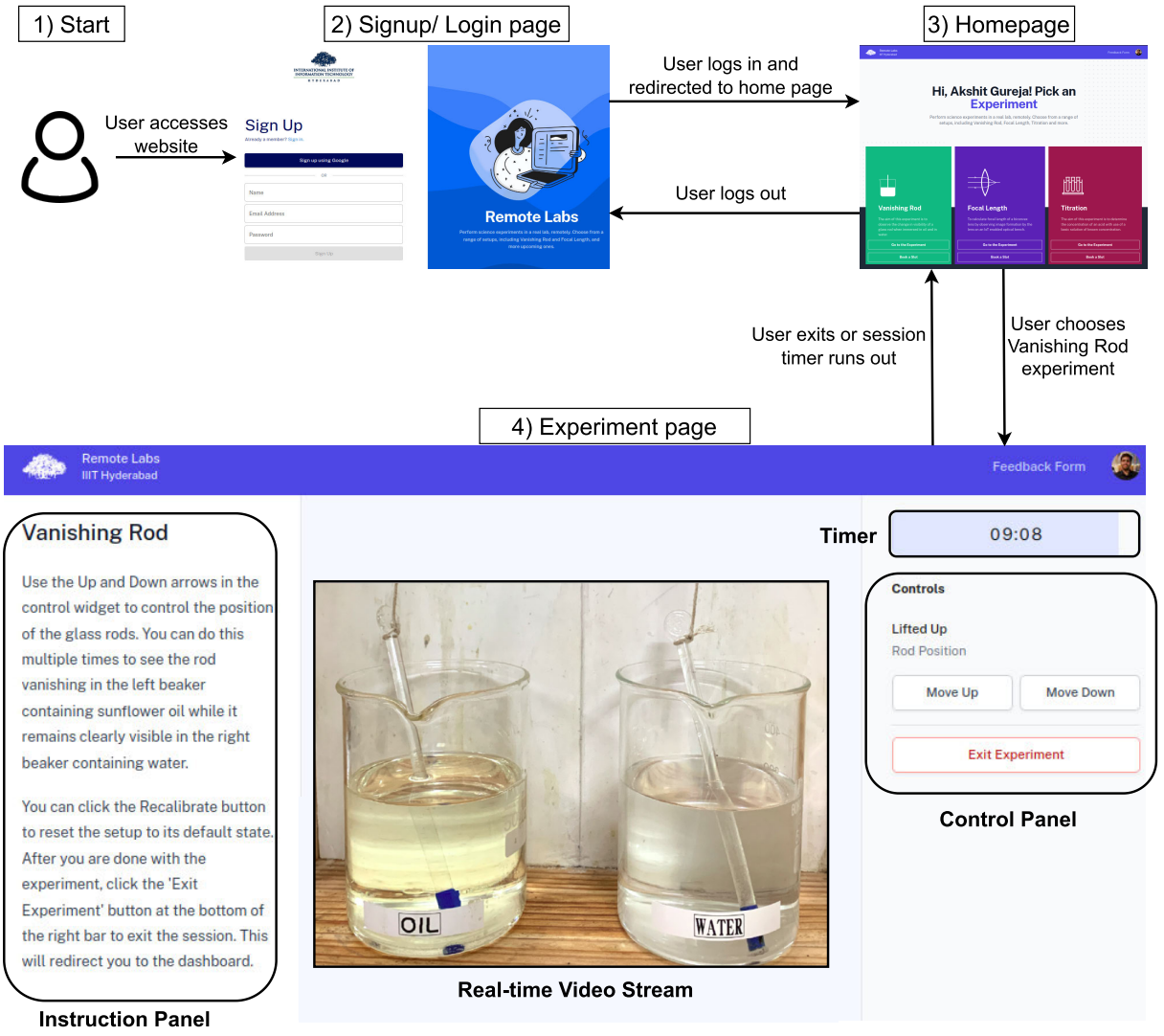


FIGURE 12. User perspective of the execution flow.

component-based JavaScript library, allows us to develop and create advanced component-based user interfaces and is required as the platform has the potential to reuse UI components for different purposes. This allows for the frontend to be highly customisable and scalable. For the purpose of verification, both Google OAuth 2.0 has been integrated, and a local authentication system has been built. The Google OAuth 2.0 allows for seamless, smooth verification of the user. Axios,[19] a promise-based HTTP(S) client library, has been used to communicate between the frontend and backend as it provides the ability to intercept HTTP(S) requests from the browser or the server.

b: P2P VIDEO STREAMING

Live results are the foundation for conducting any physical experiment. The output changes must be visible when one tweaks any inputs during experimentation, enforcing minimal capture-render display in the video streaming service [6]. To live-stream the experiment and its outputs for added user experience, the platform uses WebRTC.[20] This modern open-source protocol allows real-time communication and data-sharing between two peers with minimal latency. Peers refer to the clients who exchange any information or data using WebRTC. Every client is identified using a unique PeerID. WebRTC, is a P2P protocol which means that it connects the video streaming sources with the consumer without any server joining them. It is noteworthy to mention that WebRTC does not use a central server for relaying the streaming data; however, it does use one for the signalling phase and connecting with the other peer. Moreover, after the signalling stage is completed and the peer is discovered, the data or stream happens entirely P2P. For the purpose

[19]https://axios-http.com/

[20]https://webrtc.org/

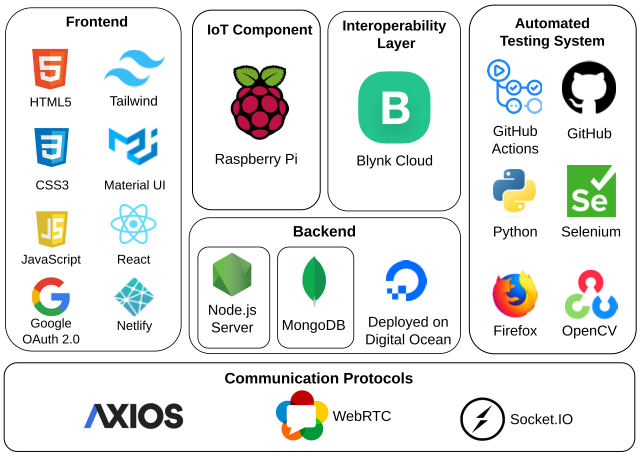


FIGURE 13. Technologies enabling RLabs platform.

of video streaming, an open-source implementation of the WebRTC protocol has been used, which uses the Node.js-based PeerJS server for the initial brokerage phase and the signalling to happen. The implementation captures the local stream of the experiments from the RaspiCam and the unique PeerID corresponding to the Raspberry Pi attached to the experiments. The P2P connection happens only when the user wishes to perform any experiment. WebRTC for our implementation has shown to have a latency of 200-400ms for one-way communication on a Raspberry Pi 3B+.

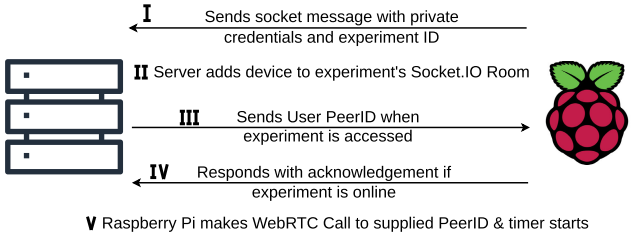


FIGURE 14. WebSocket communication between Raspberry Pi and server.

2) BACKEND SERVER

Using the MERN stack provides Express.js, a server-side web framework based on Node.js, which handles URL routing and asynchronous HTTP requests and responses. This helps connect with MongoDB, providing a scalable, document-based database to store all user credentials, experimental data, tokens, and environment variables. With a powerful Node.js-based backend server, the platform can be accessed easily once a user signs up, which can be done either locally or by connecting their Google accounts. All the authorization and verification details of the accounts signed up with the platform are stored securely in the MongoDB database. Only authorized clients are permitted to access the platform and use the hardware setups. Having a Node.js server makes our web application more user-friendly and accessible, as the features and implementation would depend on the server's node version, irrespective of the user's browser. The RLabs platform operates using a client-server connection established to access the experiments. The client-server communication utilises HTTP(S) with synchronous object transfer calls for transmitting and receiving data based on JSON using REST APIs. The asynchronous data flow is handled using "Asynchronous JavaScript and XML (AJAX) [34]," whereas the communication to the hardware setups takes place via Blynk. Hence, the server is tasked with handling communication with both the client and the Raspberry Pis, acting as an intermediate between them. The server and database have been hosted as a droplet on Digital Ocean for a paid annual subscription.

3) BLYNK CLOUD

Forming the interoperability component, Blynk, the IoT platform provides services that are widely used to control hardware remotely, store and display remotely collected data. Blynk, through its developer mode,[21] provides clients with the ability to modify and receive values using REST APIs from the cloud. It is this very feature which has been used extensively to communicate values to the hardware. When users try accessing and conducting a particular experiment, clients are connected to their desired experiment via Blynk. On providing the desired inputs using the platform, the values are modified on the corresponding Blynk device using REST APIs, which are then communicated to the Raspberry Pis of the corresponding experiments using Blynk's proprietary protocol.

4) IoT COMPONENT:

The IoT component comprises the hardware nodes, made of Raspberry Pi, sensors and actuators which facilitate remote experimentation. It also includes the interaction of Raspberry Pi with the experiment frontend for WebRTC video streaming, the Blynk cloud using its proprietary protocol and the WebSocket communication with the backend server. The WebSocket API enables bidirectional communication between a client and a server. The RLabs platform uses Socket.IO[22] - a library built on top of the WebSocket protocol. It provides additional guarantees like fallback to HTTP long-polling. Once a Raspberry Pi opens the corresponding web-page intended for communications with the setups, a Socket.IO connection is established between it and the server. As depicted in Fig. 14, after initialising the connection, the Raspberry Pi sends a socket message to the server with private credentials and its corresponding experiment's ID to gain authorisation (I). Once a Raspberry Pi is authorised, the server adds the device to a Socket.IO room (II) corresponding to the experiment it is responsible for. These rooms facilitate sending messages in bulk to each Raspberry Pi associated with a given experiment. When a user makes a request to

[21] https://docs.blynk.io/en/concepts/developer-mode
[22] https://socket.io

the server to begin an experiment session, post-checking the occupancy and availability of the experiment, the server sends a socket message indicating the same to every Raspberry Pi associated with the experiment (III). This message contains the user's PeerID for WebRTC communication. Once a Raspberry Pi receives a user's credentials, it responds to the server with an acknowledgement message (IV). This informs the server that the experiment is online and unoccupied. Once the Raspberry Pi sends an acknowledgement, it makes a WebRTC call (V) to the user using the supplied PeerID and simultaneously starts a timer for the experiment session. When the allotted time for the session runs out, the Raspberry Pi ends the call and sends a socket message to the server indicating the same. When the server receives a socket message corresponding to a session's end, it updates relevant records and marks the experiment as unoccupied.

The RLabs platform proposed in this paper is a globally deployed web platform, allowing everyone with universal access to use the hardware setups of the above-mentioned experiments remotely on any smart device capable of opening a webpage on a modern browser, requiring no supplementary applications to be downloaded. As mentioned previously, the platform's frontend is hosted for free on Netlify[23] and the backend server and database hosted on Digital Ocean[24] with a paid Blynk subscription. Moreover, implementing the platform as four separate components makes the platform modular. This makes the RLabs platform easier to scale as both horizontal and vertical scaling can be achieved simultaneously without affecting the other components. Fig. 13 quickly summarises all the technologies which are used for implementing RLabs. It can be observed that the RLabs platform has been implemented by integrating multiple open-source features, packages, libraries, applications and non-proprietary tools, making the platform easy to replicate. The whole pipeline for remote experimentation built upon these technologies must work properly which reflects the importance of reliability in RLabs. Hence, an automated testing system is proposed to ensure the correct functioning and working of the platform.

D. AUTOMATED TESTING SYSTEM

For a platform allowing users to remotely conduct scientific learning by doing mechanical actuation, the reliability and consistency of results is a major concern. These become even more vital considering that these experiments are utilised for educational and research purposes where accuracy is of prime importance. Hence, after the experiments are deployed, it is crucial to receive real-time updates on the connectivity, availability and working of the experiments to guarantee optimal performance and timely maintenance. These updates are obtained by performing several checks, including the working of hardware components, the functioning of cameras, the stability of network connectivity and the responsiveness of the cloud service. Various strategies for executing these checks manually have been previously proposed. However, the manual execution of these assessments can be time-consuming and inefficient, especially when they need to be performed on a daily basis.

Automated testing is a process of consistently monitoring the functionality and integrity of experiments showcased on the RLabs platform automatically without any human interference and emerges as a vital solution in this context, facilitating a streamlined, efficient, and hassle-free means of monitoring the experiments. To ensure the maintainability of the platform, including all of its components mentioned above, along with the reliability of the results, an extensive automated testing system has been developed for the RLabs platform. It is designed specifically to identify and help troubleshoot anomalies and irregularities in experimental processes effectively, ensuring the consistency of results. Fig. 15 shows that the *Web Driver, OpenCV scripts and Mail Handler* constitute the testing system, invoked periodically by the Selenium-based automation script, which is deployed on GitHub Actions. The system comprises two major testing elements - 1) Hardware system checks and 2) Computer-Vision (CV) testing. Both these elements work synchronously for a comprehensive testing system that has been automated using the GitHub Actions workflow, constituting the CI/CD component. Whenever the script detects an error, fault or inaccuracy, the mail handler is used to alert the software administrators regarding it. An elaborate discussion of the elements, components, their implementation and interaction with the RLabs platform is given below.

1) HARDWARE SYSTEM CHECKS

This element of the automated testing system primarily focuses on reporting errors that arise from the scripts running in the background of Raspberry Pi and other microcontrollers. Catch handlers are integrated into the scripts to detect and report errors at various stages. On the hardware side, errors are identified, including those thrown by hardware packages controlling the stepper motor drivers and instances of crossing set thresholds. Hardware assets, including moving components like motors, are also recalibrated after each experiment session to guarantee precision in reporting distances. For instance, the Focal Length experiment is equipped with limit switches that facilitate the recalibration process. Suppose an error is detected in the Raspberry Pi's working or the extended IoT system. In that case, the Raspberry Pi updates a variable on the Blynk Cloud with a pre-decided error code pertaining to the issue or fault that has occurred, which would then be captured by the GitHub workflow described below.

2) COMPUTER-VISION TESTING

The hardware system checks described above offer the capability to capture only the errors and faults that might occur in the hardware nodes. As mentioned earlier, the testing system aims to ensure that any anomalies or

[23]https://www.netlify.com/
[24]https://www.digitalocean.com/

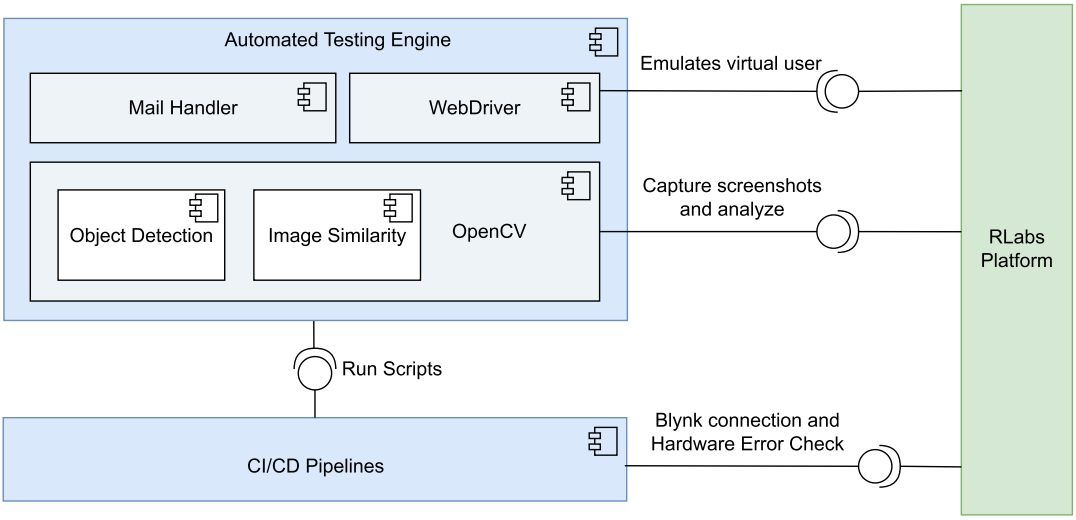


FIGURE 15. Components of the automated testing system.

operational inefficiencies are promptly identified, reported and addressed. For the aim of comprehensive testing, there should be a constant check on the software platform's working along with the hardware's functionality. This is made possible by the OpenCV-based script, which is invoked at regular intervals, typically every eight hours. This comprehensive checking script, although requiring the temporary halting of the experiment, ensures all components are working as intended and uses CV for monitoring mechanical changes such as changes in the position of components, etc.

Both experiments aim to detect and track moving objects, such as glass rods for the Vanishing Rod experiment and object and screen platforms for the Focal Length experiment. This task can be considered an extension of foreground detection, where moving objects are considered in the foreground. Various methods exist to monitor these changes depending on the specific use case, such as Deep Learning-based tracking or a combination of image processing techniques like background subtraction, optical flow, and morphological transformations. Each approach has its own set of strengths and weaknesses, including computational intensity, real-time performance, accuracy, and robustness in different environmental conditions. The system implemented in this work employs background subtraction and morphological transformations [35] due to their lower computational intensity and ability to provide real-time results. Figure 16 illustrates the algorithmic pipeline of the image processing and CV techniques used in the Focal Length experiment. The techniques used are: *Background Subtraction*, which eliminates the background from screenshots, emphasising only the moving objects and the platforms. Following this, *Morphological Transformation* is applied, particularly the 'closing' technique (a combination of dilation and erosion), to reduce image noise and fill small spaces, aiding in more accurate foreground segmentation. Next, *Image Filtering* through median filtering further reduces noise, enhancing image quality. The final step, *Image Thresholding*, involves thresholding the image to enable the drawing of bounding boxes around these foreground objects, completing the process.

Once the bounding boxes are annotated over the object and screen platforms, their locations can be tracked to determine their movements. This helps check if the motors are working correctly and moving to the desired location. In the Vanishing Rod experiment, the system tracks the movement of glass rods, as mentioned above. The experiment's motors are tested by moving the rods up and down, and the data obtained is used to verify their functionality. The beaker's water level is also monitored using Structural Similarity Index Measure (SSIM) [36]. This is crucial as the water in the beaker can evaporate over time. In the Focal Length experiment, Fig. 16 illustrates the usage of the CV pipeline. The system moves the object and screen platforms and tracks the distance travelled by the platforms. Since the platforms move linearly and the total length of the setup is already determined, this fact is used to verify if the platforms have moved to their desired locations, thus verifying the efficiency of the motors.

E. EXECUTION FLOW FOR AUTOMATED TESTING

As mentioned earlier, a Selenium script deployed on GitHub Actions is used to automate the whole testing process. Selenium is a widely used open-source web browser automation software framework, enables interactions with web applications programmatically, such as clicking buttons, filling forms and navigating web pages. The Selenium script emulates a virtual user that logs into the RLabs platform and assesses the experiment's functioning by manipulating the controls on the dashboard. Fig. 17 shows that the script deployed on GitHub Actions as a workflow runs periodically (after every eight hours) and starts the Selenium

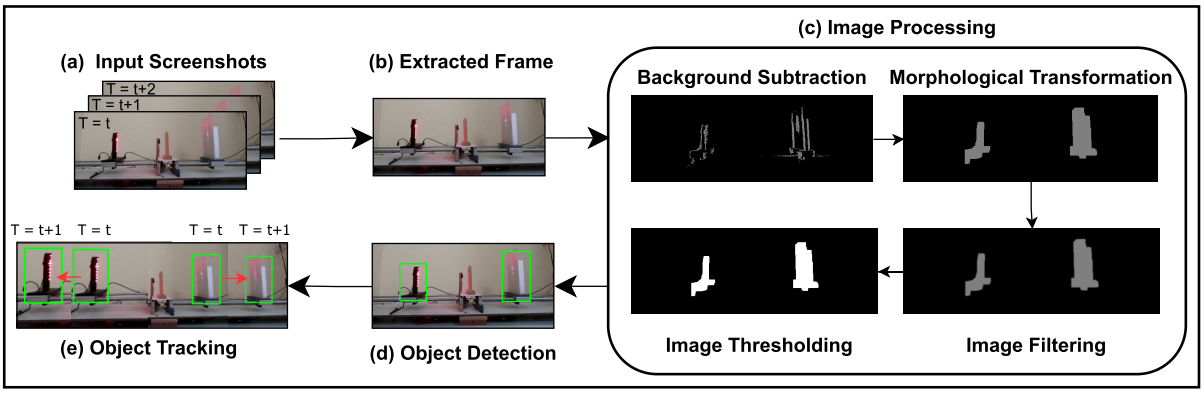


FIGURE 16. Algorithmic pipeline of the CV script for testing focal length experiment.

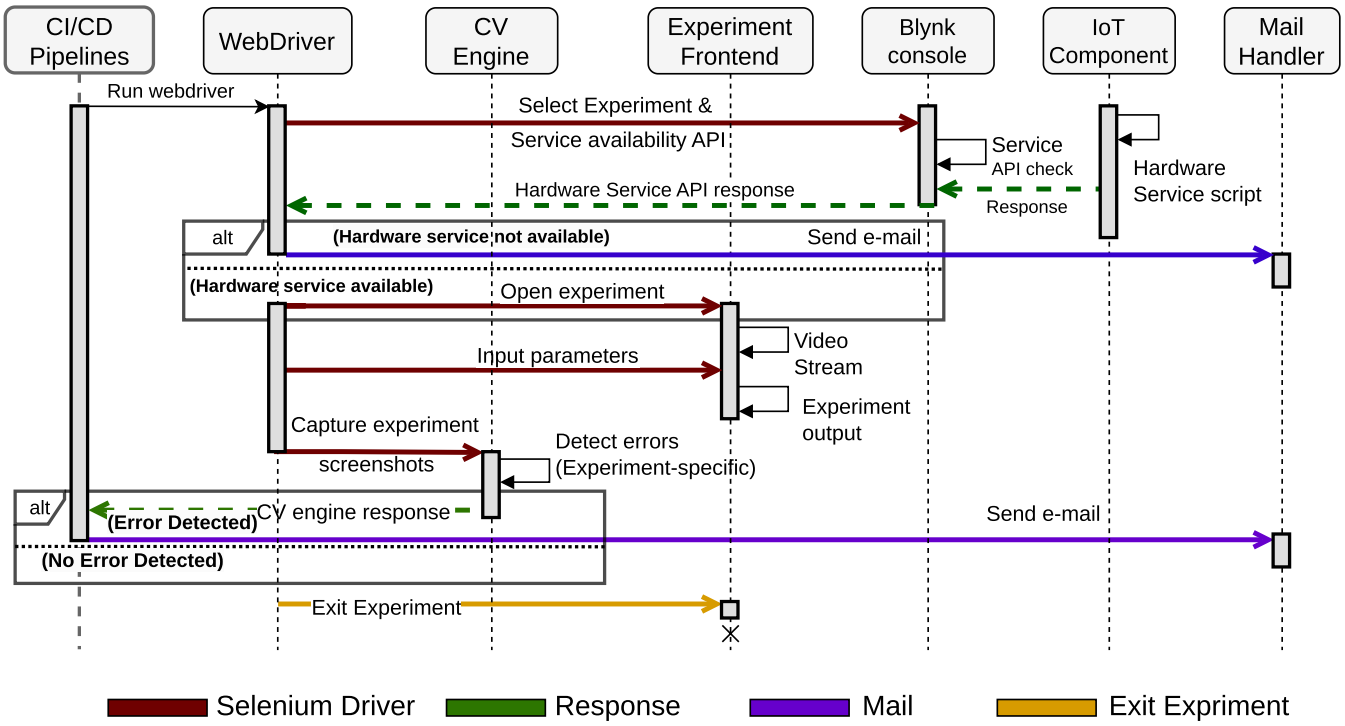


FIGURE 17. Automated testing sequence diagram.

driver to emulate a virtual user that logs into the RLabs platform, selects an experiment to perform and assesses that experiment's functioning by checking the video stream and manipulating the input parameter controls. It initially awaits for the video stream to begin and captures screenshots of the experiment's initial state. Then, the input parameters are given to prompt actuation in the experiment, and multiple screenshots are then captured, which are processed by the CV engine that employs the techniques described earlier to detect if there is any error. Please note that the video stream and the experiment output are obtained on the experiment frontend by following the execution described previously in Fig. 11. Hence, the detailed flow for these has been omitted from the sequence diagram for automated testing.

Simultaneously, the GitHub workflow also makes REST API calls to the Blynk Cloud to check if the experiment's hardware is connected to Blynk and if any error code was received from the hardware setups. The workflow receives an appropriate HTTP response. If there is an error raised by the CV engine or the hardware is found to be disconnected or an error code was transmitted from the hardware as part of the system check, an appropriate mail is sent by the mail handler notifying the administrators and the software team regarding the same. In case of perfect remote experimentation, the Selenium driver simply logs out. This sequential flow, as mentioned earlier, is followed every eight hours. As the rules for anomaly detection are highly specific and tailored to the nature of each experiment, particularly the CV techniques,

there is a separate testing framework for each experiment with a distinct CV engine. The two scripts for the Vanishing Rod and Focal Length experiment are hosted on the Github Cloud Server and are triggered at predefined intervals by cron jobs, ensuring regular and automated testing.

F. EVALUATING THE RLABS SOFTWARE SOLUTION

It is essential to look at some metrics demonstrating the system's performance capabilities and reliability to provide a better foundation for the RLabs software solution. This further aligns with the overall goal of proposing an affordable and scalable remote lab. The RLabs platform and the automated testing suite have been evaluated for their performance below.

1) RLabs PLATFORM

As RLabs solution aims to provide remote access to scientific experiments, the platform must allow real-time interaction and be scalable. Considering this mandate, the software platform needs to have low latency. Hence, the latency of the platform during user experimentation and for the WebRTC video streaming are discussed in detail below.

a: USER EXPERIMENTATION

Latency during user experimentation refers to the complete one-way latency, that is, the time taken for an input provided by the user on the platform's frontend to reach the Raspberry Pi in the deployed hardware via the backend server and the Blynk cloud. Only one-way latency is considered because the response time from the hardware to the user depends on the experiment and its functioning. Thus, the response time from hardware to the user varies from experiment to experiment. The average one-way latency for a single user is measured to be approximately 220 ms.

b: WebRTC VIDEO STREAMING

WebRTC video streaming in our implementation gives a latency between 200-400 ms and an average bandwidth usage of 1.9 Mbps during the P2P streaming. WebRTC-internals is an internal chromium tab that stores the statistics for an ongoing WebRTC session and is used to measure the same.

The observations above are for a single user conducting remote experimentation on the RLabs platform. As more users start accessing the platform, it is essential to discuss the scalability of RLabs from a software perspective. A fundamental requirement of building a scalable system is its consistent availability to the users, measured by uptime for a web-based platform. The RLabs platform has a 30-day average global uptime of 99.95%, as measured using the uptime check provided by Digital Ocean. Moreover, there are two primary aspects of scaling RLabs - the latency experienced per user and the backend server's ability to handle increasing load as the number of concurrent users on the platform increases. Both these aspects have been discussed in detail below.

As RLabs requires real-time interaction with the hardware, it is crucial to ensure that a user's experimentation and experience is unaffected by more concurrent users on the platform. Such analysis of how the latencies vary per user can typically be done using multi-user stress tests. However, multi-user stress tests for remote lab solutions present a unique case, given their interdependence on both hardware and software. RLabs experiments involving mechanical actuation present a case of forced-single user access, as multiple users cannot control motor actuation simultaneously. Thus, the number of people who can experiment at any given point is limited by the number of hardware nodes. Considering this limitation, the latency analysis has been conducted with five hardware setups readily available with the authors.

Fig. 18a shows how the average one-way latency experienced by each user as the number of concurrent requests on the platform increases, going up to five different users operating on five different hardware nodes simultaneously. As the figure shows, the average one-way latency across all users shows an upward trend, showing that as more users access the server, the system experiences a very slight increase in latency, most likely due to additional load processing. The vertical blue lines signify the minimum and maximum latency observed for the respective number of concurrent users. Moreover, note that the analysis doesn't include WebRTC streaming as it is a P2P video and doesn't add any latency or load on the server.

However, such latency-related analysis need testing for a larger number of users. In the context of RLabs, the hardware can be simulated as a script listening to the Blynk cloud. Fig 18b depicts the latencies observed for five simulated hardware nodes and compares them with the values obtained for the real hardware nodes. The simulation closely resembles the actual latency values and, hence, can be considered a good proxy for actual hardware nodes. Thus, the scalability for a larger number of concurrent requests can be simulated similarly. Fig. 18c shows the average latency observed per user as the number of users sending concurrent requests or inputs increases for simulated hardware nodes. It shows a gradual increase in average latency, as expected, due to the additional load of handling concurrent requests.

Discussions on scaling RLabs cannot be limited to latency, and the server's ability to handle an increasing number of users and requests must also be considered. While handling concurrent requests from users, the server might be overwhelmed after a limit. This makes it essential to look at the performance of the deployed server while concurrent users access the experiment. The server for the RLabs platform is hosted as a Digital Ocean droplet, which is a virtual private server (VPS) in the Bangalore (BLR), India region, with the Ubuntu 22.04 operating system installed on the droplet. The hosted droplet has the following configurations:

1) Memory: There is 1GB memory allocated to the Droplet, of which an average of 61.9% is used

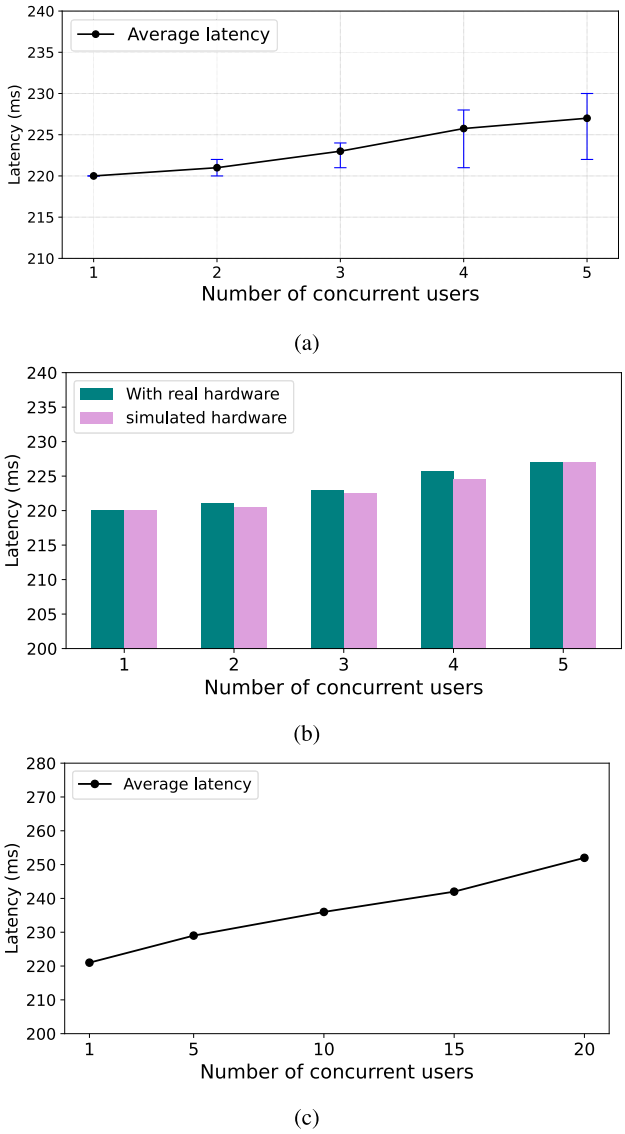


FIGURE 18. One-way latency (a) For five concurrent users on real hardware node (b) Comparison of values obtained with real hardware nodes and simulated hardware (c) Average latency for upto 20 simulated concurrent users.

currently. The low memory usage depicts that the application is lightweight.

2) vCPU: The droplet currently has access to 1 Intel virtual CPU core and uses 1-2% of the available processing power at any given point in time.
3) Disk Space: The droplet has an allocated storage capacity of 25 GB.

Fig. 19 shows the incoming and outgoing bandwidth usage in KB/s by the server's network interface. It also shows the CPU usage (in %) with increased concurrent user inputs (up to five for five currently available hardware nodes). Moreover, these values depict the server's performance while the concurrent requests are being processed. Zero users refer to the rest state of the server while there are no active users on the platform.

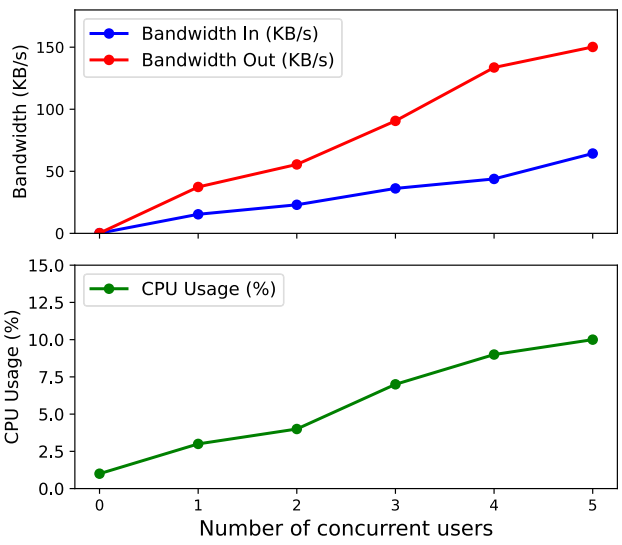


FIGURE 19. Server performance metrics for increasing concurrent users.

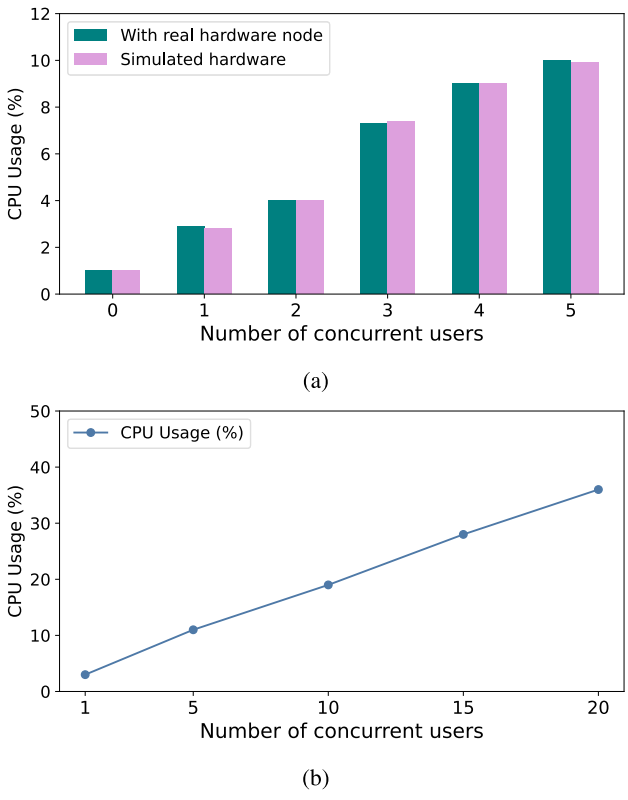


FIGURE 20. CPU usage (a) Comparison of values obtained for real hardware node and simulated hardware (b) CPU usage for 20 simulated concurrent users.

The CPU usage for a single core, as observed, rarely exceeds 10% in practical use scenarios, with around 90% of the CPU being idle at all times. Moreover, as done previously for latency analysis, we can simulate more concurrent users accessing hardware nodes through a script. The server's CPU usage for five concurrent requests for five simulated

hardware nodes and its comparison with real hardware node also given in Fig. 20a As expected, our simulation of a hardware node provides a good approximation of how the CPU usage would scale. Fig. 20b presents the CPU usage (in %) for upto 20 concurrent users on the platform by simulating 20 hardware nodes. As can be observed, the CPU reports a usage of around 36% for 20 concurrent requests. By direct extrapolation, it can be said that the server would be able to handle 50 concurrent requests every second. The server's high performance, even on a single core, is evident from the fact that it is involved with only the relaying of the input parameters and the output values, with no video element. This makes the server lightweight in its implementation and can thus easily handle a large load of users conducting experiments through the platform. Furthermore, to handle an even larger number of concurrent requests, the number of CPU cores can be increased.

2) AUTOMATED TESTING SYSTEM

As mentioned earlier, the automated testing system emulates a user and performs the experiment by using GitHub Actions workflow as the CI/CD component. With every experiment node hosted on the platform to be tested, scalability of any platform requires the automated testing system to be short and efficient. Furthermore, deployment processes, resource utilization and response time to address any issue are all dependent on the time taken by the automated testing system. For such analysis, the detailed execution flow given above can be broadly divided into three steps or stages:

1) Installation and Setup: Every time the script restarts to perform the automated testing, it installs the runtime libraries.
2) User Emulation and Experimentation: This refers to the selenium script choosing the experiment, providing input parameters and taking screenshots.
3) Post Processing: This refers to the execution stage where the screenshots taken are analyzed using the CV methodology mentioned earlier which varies according to each experiment.

Table 4 shows the time taken to perform each of these steps for the Vanishing Rod (VR) and Focal Length (FL) use case experiments.

As depicted in Table 4, the user emulation and experimentation depend on the nature of the experiment. Given focal length has, by virtue, a more complex actuation, it is bound to take more time to test completely. Moreover, the post-processing, as described above, depends on the CV pipeline to automatically test the system and hence is also an experiment-dependent component.

G. ADDITIONAL FEATURES OF RLABS PLATFORM

The RLabs platform and testing system discussed above together form an exhaustive and comprehensive solution for conducting scientific experiments remotely. However, any system or platform developed for public use also needs to pay close attention to its performance and potential to scale and provide an enhanced user experience. Multiple features and crucial functionalities, focussing on such aspects, have been integrated into the platform, and their elaborate explanation has been given below.

TABLE 4. Automated testing metrics.

Actions/Tasks	Time Taken	
	VR	FL
Installation and Setup	38s	38s
User emulation and experimentation	1m 54s	2m 45s
Post-processing	2s	7s
Total	**2m 34s**	**3m 16s**

1) USER MANAGEMENT ON RLABS PLATFORM:

As mentioned earlier, RLabs experiments involve mechanical actuation and force single-user access. To resolve issues with user accessibility to the experiments and manage the supply and demand for experimentation in such scenarios, methods such as hardware multiplexing, queues and slot booking must be implemented. While hardware multiplexing provides a supply-side solution, queues and slot-booking are software implementations that work on the demand-side of the pipeline enabling remote experimentation.

Hardware multiplexing refers to the platform's ability to accommodate hardware horizontal scaling and handle the increasing nodes of an experiment by redirecting users to different nodes of the same experiment. Horizontal scaling is one of the most efficient ways to provide more user experimentation instances available. Due to the mechanical nature of most of the experiments, there must be single-user access to a particular hardware node. However, by horizontal scaling, multiple users can access multiple experiment instances simultaneously while the platform adheres to the single-user access limitation. This makes hardware instance expansion critical and the ease of integrating multiple instances, once built, into the platform with a few clicks. For adding new instances of the experiments whose dashboard and data presentation formats are pre-designed, the administrators require only the Blynk authentication tokens and the WebRTC initiating experiment room IDs. This horizontal scaling by creating more hardware instances is an example of supply-side management. So, if a user wants to perform a desired experiment and some nodes are already occupied by the other online users, to maintain the single-user access, the student would automatically be given access to another hardware node which is live, running, connected to the platform and currently vacant in terms of usage. The single-user access, along with hardware multiplexing is depicted in Fig. 21a.

Queues are a software implementation that is an improvement on the demand-side that forces First-Come, First-Serve access or is referred to as the First-In, First-Out (FIFO) scheduling strategy. Queues are used to give sequential access to the users in the order of when the demand to access was recorded. As depicted in Fig. 21b, if User 1 is already

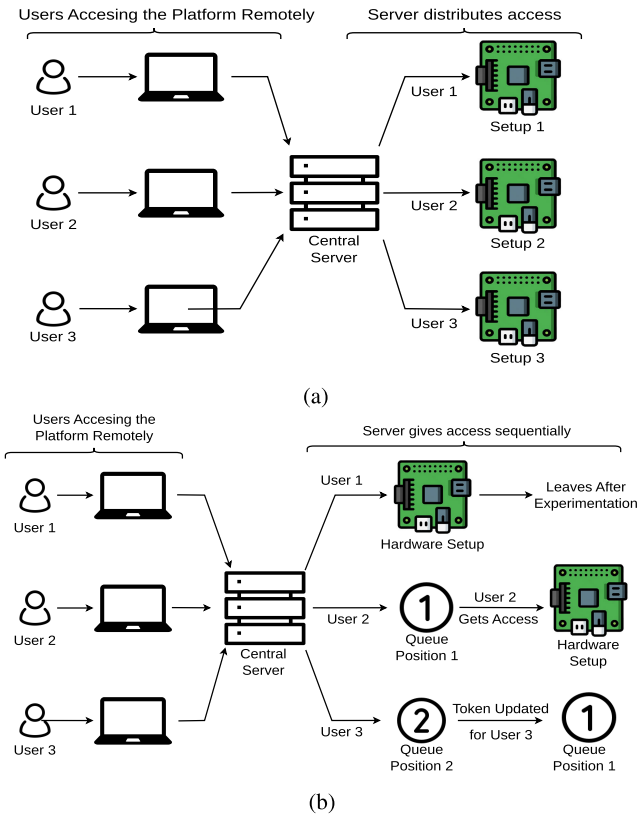


FIGURE 21. User management in RLabs platform. (a) Hardware multiplexing with 3 hardware nodes of the same experiment. (b) Queue implementation.

using an experiment and the second user wants to access, they are put in a queue with an incrementing waiting queue token number. When the first user leaves and the experiment is free to use again, User 2, who is already in the queue, gets the first chance to experiment, and the token numbers reduce for all the users in the waiting queue for that particular experiment. A combination of hardware multiplexing and queues can handle a large load efficiently. Slot Bookings are another widespread demand-side management software implementation. This refers to the platform's ability to distribute the user demand of accessing hardware nodes over slots of predecided time intervals. Users can reserve and book an experiment for a particular time duration to perform remote experimentation. Such slot bookings can also be used to resolve high demands on such remote lab solutions.

2) MULTIPLE VIDEO STREAMS FOR THE SAME EXPERIMENT

The software platform must also be compatible to support multiple camera streams for a single experiment. The Focal Length experiment explained above provides two views of the experiment's hardware node - a closeup view of the image on the screen and a side-view of the entire experiment's setup. Using a combination of WebRTC and Socket.IO-based streaming, the server adds all the Raspberry Pis pertaining to a single hardware instance to a dedicated Socket.IO room post-authorisation. When a user tries accessing a hardware instance, server post-user authentication sends a socket message with the user PeerID to each Raspberry Pi in the dedicated room of the desired experiment, after which each Raspberry Pi initiates a WebRTC call individually. This seemingly complex pipeline allows RLabs to have numerous cameras if required by an experiment.

V. RESULTS

This section presents the performance of the proposed RLabs system based on the NFAs proposed earlier. This will include cost tables for building the experiments to demonstrate their affordability, assembly steps for the miniaturised setups that make the experiments portable, compatibility of the RLabs system with different hardware boards, IoT platforms and cameras, availability of the deployed experiments, and an analysis of the user survey collected.

A. LOW-COST HARDWARE SETUPS

Table 5 shows the total material costs for building the lab-scale and miniaturised setups for the two use case experiments in INR. It can be observed that the total cost of the lab-scale Vanishing Rod and Focal Length experiments are 7700 INR (92 USD[25]) and 14500 INR (174 USD), respectively. For the miniaturised Vanishing Rod and Focal Length experiments, the total cost of the setups was 4500 INR (54 USD) and 11000 INR (132 USD). There is a 41 % and 24 % reduction in costs for making the miniaturised setups for Vanishing Rod and Focal Length experiments, respectively. Firstly, the setups are low-cost and can be attributed to using single-board computers, which has brought down the costs as individual servers or PCs are not required for hosting the experiments. Raspberry Pi 3B+ (5000 INR) is the costliest item (33 % and 66 %) in the lab-scale setups. This cost can be reduced by replacing it with cheaper boards like Raspberry Pi Zero 2 W (1650 INR), significantly reducing the costs observed in the miniaturised setups.

B. PORTABLE HARDWARE SETUPS

To showcase the portability of the built experiments, the mass and volume of the experiments are reported along with their step-by-step assembly. Table 6 presents a comparative analysis of the lab-scale and miniaturised setups, focusing on mass (excluding the beakers filled with oil and water) and volume. There has been a 5.4 and 12-times reduction in the weight and volume of the Vanishing Rod experiment from the lab-scale to the miniaturised setup. Similarly, in the case of the Focal Length experiment, the weight and volume were reduced by a factor of 2.8 and 4.4, respectively.

Figs. 22 and 23 show the step-by-step assembly of the two use case experiments from their components in a modular fashion. The setups are designed using various techniques to make their assembly easy. The parts have different shapes that fit together, making it intuitive for the user to join them. This

[25] 1 USD is approximately 83 INR as of December 2023.

TABLE 5. Cost tables for procuring the components of the two use case experiments.

(a) Costs for lab-scale Vanishing Rod setup

Lab-Scale VR		
Item	**Qty**	**Cost**
Raspberry Pi 3B+	1	5000
28BYJ-48	2	160
ULN2003	2	80
Body Frame	1	500
Camera	1	500
Glass Rod	2	60
Beaker	2	400
Misc	1	1000
Total		**7700 INR 92 USD**

(b) Costs for miniaturised Vanishing Rod setup

Miniature VR		
Item	**Qty**	**Cost**
Raspberry Pi Zero 2W	1	1650
Body Frame	1	1000
Camera	1	500
Fabricated PCB	1	300
28BYJ-48	1	80
ULN2003	1	40
ESP32	1	400
Glass Rod	2	30
Beaker	2	100
Misc	1	400
Total		**4500 INR 54 USD**

(c) Costs for lab-scale Focal Length setup

Lab-Scale FL		
Item	**Qty**	**Cost**
Raspberry Pi 3B+	1	5000
NEMA 17	2	1700
A4988	2	300
USB camera	1	1200
RaspiCam	1	500
8mm Axle	4	1200
Axle support	8	400
Screw Rod + Nut	2	650
Shaft coupler	2	150
Bearing	2	200
Slider	4	800
3D printed parts	1	300
Wooden Planks	1	600
Misc	1	1500
Total		**14500 INR 174 USD**

(d) Costs for miniaturised Focal Length setup

Miniature FL		
Item	**Qty**	**Cost**
Raspberry Pi zero 2W	2	3300
NEMA 17	2	1700
A4988	2	300
Camera	2	1000
Support Wheels	4	400
Aluminium Profile	2	800
Timing Belt	2	200
Pulleys	2	150
Idler	2	150
3D printed parts	1	1500
Fabricated PCB	1	500
Misc	1	1000
Total		**11000 INR 132 USD**

TABLE 6. Comparison of lab-scale and miniaturised experimental setups.

Type of Setup	Lab-scale setup		Miniaturised setup	
	VR	FL	VR	FL
Mass (KG)	1.8	6.2	0.33	2.2
Volume (cm^3)	30×20×32	25×150×20	10×10×16	10×85×20

also eliminates the use of screws and bolts to some extent. For example, in the Vanishing Rod setup, the exoskeleton has a specially designed mount for the motor that exactly fits over the motor and holds it without any screws. Similarly, sliding nuts and the aluminium profile create a slide and lock mechanism for the Focal Length setup. The sliding nuts easily slide into the profile slots and can be tightened using a bolt that holds the centre base and other parts against the profile. This system overcomes the need to drill holes into the profile, making it easier for an individual to assemble the setup. Both setups are entirely made of off-the-shelf parts along with 3D-printed modules that are usually available in the local hardware and electronics stores. The combination of modularity and miniaturisation renders the setups portable, similar to IKEA[26] products, enabling users to assemble and disassemble them easily. This portability facilitates the shipment of setups to various institutions and universities worldwide, including remote rural areas, thereby extending their accessibility and making them easier to deploy and setup.

C. COMPATIBILITY WITH DIFFERENT HARDWARE BOARDS, CAMERAS AND IoT PLATFORMS

In order to ensure broader compatibility, the software architecture is designed to focus on supporting a wide range of hardware boards. This is shown using different boards (such as Raspberry Pi and ESP32) to implement use case experiments. The boards used must be able to connect online using any IoT platform that facilitates data access via APIs. This allows for the accommodation of various hardware boards. However, there can be boards that do not support internet connectivity. Those boards can then be coupled with an ESP8266 or ESP32, which provides smooth data transfer from the experiment to the dashboard at an economical price point, starting from as low as 400 INR (∼5 USD).

If a board does not support live-streaming from a camera, a Raspberry Pi Zero 2 W,[27] which costs around 1600 INR (∼20 USD), can be used alongside standard RaspiCams

[26] https://www.ikea.com/

[27] https://www.raspberrypi.com/products/raspberry-pi-zero-2-w/

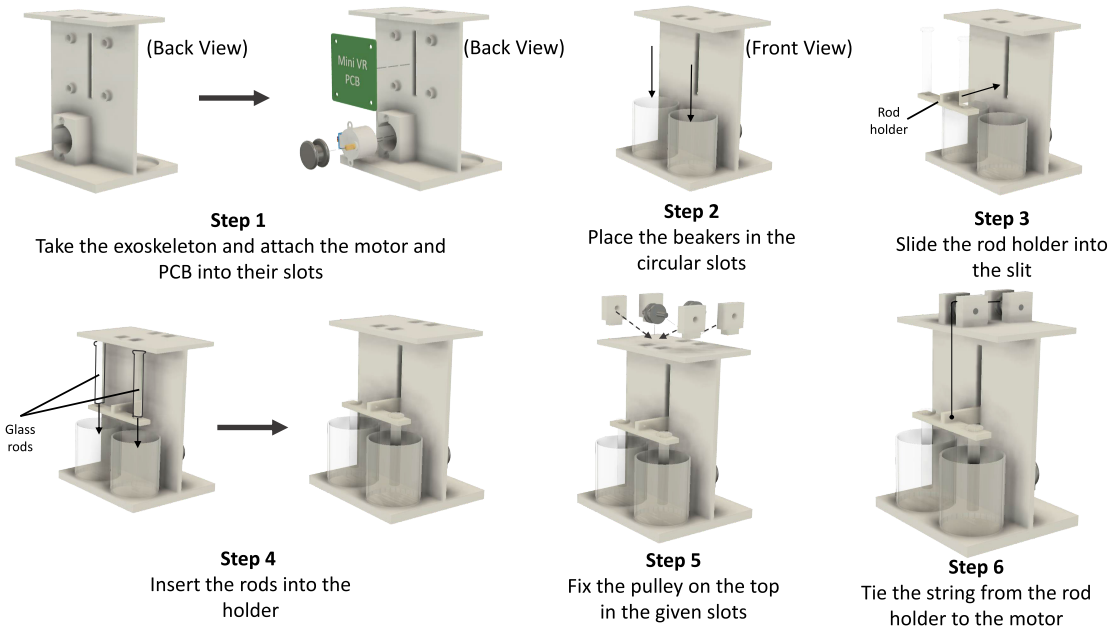


FIGURE 22. Assembly of the miniature Vanishing Rod setup.

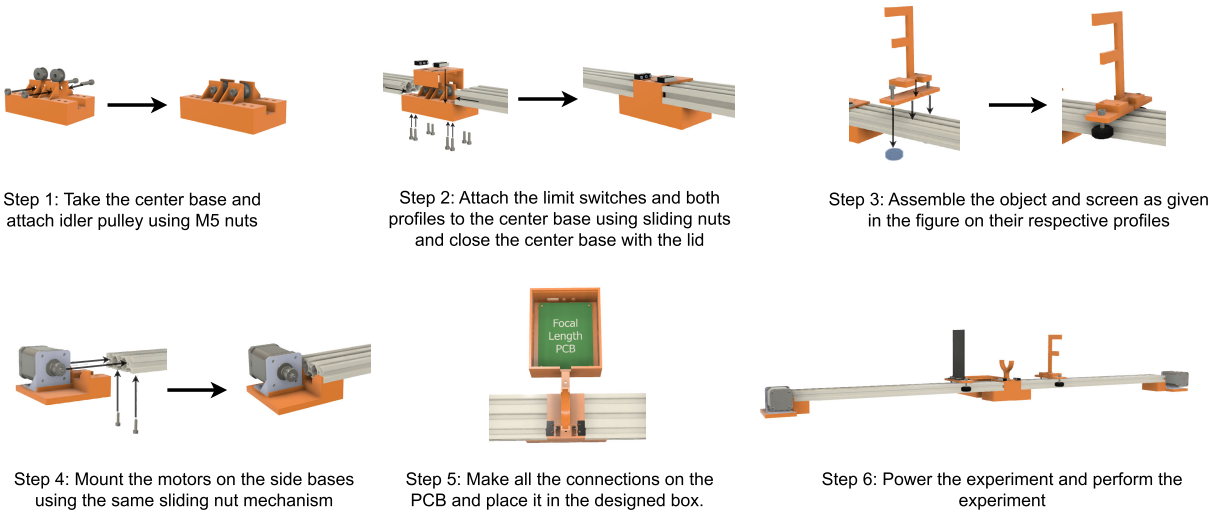


FIGURE 23. Assembly of the miniature Focal Length Setup.

solely for streaming purposes. For experiments that use a Raspberry Pi, any IP camera or USB camera compatible with and recognised by a Raspberry Pi can be used for streaming. Standalone IP cameras can be easily set up by users with minor configurations, including setting up a static IP for the camera and port forwarding to grant global access. Table 7 shows the observed delay in live-streams across different devices. The WebRTC facilitates the streaming in Raspberry Pis, while the stream from the TP-Link Tapo C100 IP Camera[28] is accessed using Real Time Streaming Protocol (RTSP). The Raspberry Pi Zero W[29] was evaluated with the streaming script as well. However, it delivered a poor-quality stream and exhibited significant lag, with delays exceeding 3 seconds. Alternatively, the ESP32 Cam module offers streaming capabilities, requiring minor tweaks to support WebRTC [37]. However, it is essential to note that the authors have not officially tested this configuration.

Once the hardware experiments are built, the next step is their integration with the RLabs platform. The Blynk cloud service, an IoT platform currently supporting HTTP(S), is used for this. However, the RLabs platform can handle other protocols from various IoT platforms, provided they allow reading and modification of variable states essential for designing experiments. To test this, the dashboard for the Focal Length experiment is created and tested on two IoT

[28] https://www.tapo.com/in/product/smart-camera/tapo-c100/
[29] https://www.raspberrypi.com/products/raspberry-pi-zero-w/

platforms: Blynk using HTTPS and Thingspeak[30] via MQTT. Other notable IoT platforms include Arduino IoT Cloud,[31] and AWS IoT Core.[32] These platforms extend their support to multiple hardware boards, including Raspberry Pi and ESP boards, through specialised libraries and packages, making integration even simpler.

TABLE 7. Latency in live-streaming of video across various cameras.

Camera	Delay (in sec)
RaspiCam + Raspberry Pi 3B+	0.35
RaspiCam + Raspberry Pi Zero 2 W	0.86
TP-Link Tapo C100 IP cam	2.01

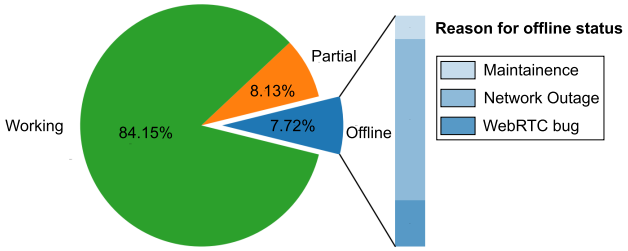


FIGURE 24. Availability of the experiments.

D. AVAILABILITY OF THE EXPERIMENTS

Fig. 24 displays the operational status of the experiments recorded by the aforementioned automated testing suite. Daily, over four months from July to October 2023, the operational status of the lab-scale experiments was logged.

The findings have been segmented into three distinct categories:

1) Online: Indicates that both the experiment's hardware and the live-stream are functioning optimally without disruptions throughout all the readings taken for the day.
2) Partial: Signifies either a malfunction in the live-stream service or an issue with the experiment's hardware at least once during the day.
3) Offline: Specifies that the experiment's hardware is entirely non-operational for the day.

Out of the 123 days (4 months), the Vanishing Rod experiment was online for 106 days, partially working for eight days, and entirely offline for nine days. The Focal Length experiment was online for 101 days, partially working for 12 days and entirely offline for 10 days. In July, the experiments generally ran smoothly, with a few exceptions, such as when an issue with the implementation of WebRTC caused the experiments to appear offline. This issue was identified and resolved within a few days. The experiments then operated without further problems. However, towards the latter part of August and early September, the labs experienced significant downtime, primarily due to a severe disruption in the campus network where the experiments were hosted. This outage persisted for nearly seven days before being rectified. There were a few instances of experiments disconnecting from the Wi-Fi network and failing to reconnect. The exact cause has yet to be determined, but it may be related to Wi-Fi router-specific issues [38]. This record-keeping has provided invaluable insights into the system's reliability, serving as a critical measure of its performance. By understanding these patterns, proactive steps can be taken to enhance the system's reliability.

E. USER FEEDBACK

A user survey was conducted, and feedback from a group of forty-five grade-9 students at Shikhar Educare, Amravati, Maharashtra, India was collected. The two use case experiments are part of their curriculum. The two experiments were delivered to and accessed by the students through a blended learning methodology where traditional in-class instruction was combined with the RLabs platform as the digital learning tool. Table 7 displays the specific questions posed and the average scores for each. The survey included standard questions [39], [40] that focused on pedagogy, usability, and learning outcomes. Additionally, users were asked to rate aspects such as the responsiveness of the experiments, live-streaming quality, and the intuitiveness of the software platform, among others. Responses were provided using a Likert scale, where a score of 1 represented strong disagreement, and 5 indicated strong agreement.

The user feedback on the remote lab solution, as shown in Table 8, indicates a positive user experience, with high scores in areas related to usability, clarity of streamed results, and content quality, suggesting that the platform is both user-friendly and educationally effective. The scores ranged from 4.09 to 4.64 on a 5-point scale, with an overall average of 4.34. A low standard deviation was observed in the student's feedback. This indicates consistency in responses, balanced engagement and reasonable agreement without excessive outliers or strong polarization. Moreover, Cronbach's Alpha[33] was used to assess the internal consistency of the responses across questions and was found to be 0.89. This implies that the questionnaire consistently captured the student's attitude towards the RLabs platform. Students felt that the remote lab was easy to navigate and use, receiving the highest score. Most users also found the experiment parameters and feedback form sufficient, reinforcing the solution's comprehensiveness and effectiveness. The users have requested to add other experiments that could complement their understanding in their comments collected at the end. The results overall suggest that the remote lab solution successfully meets user expectations in various vital aspects. However, minor improvements, like adding experiments from different subjects, could enhance the experience further.

[30] https://thingspeak.com/
[31] https://docs.arduino.cc/arduino-cloud/
[32] https://aws.amazon.com/iot-core/
[33] https://statisticsbyjim.com/basics/cronbachs-alpha/

TABLE 8. Average scores and responses of user feedback on remote labs.

Questions	Average Score	Standard Deviation
The remote lab significantly helped in my learning process	4.09	0.67
I felt engaged while using the remote lab	4.27	0.52
Using the remote lab felt similar to using a physical lab	4.24	0.52
The remote lab is easy to navigate and use	4.64	0.70
The quality of the live video stream was excellent	4.27	0.56
The experiments in the live-stream were highly responsive	4.40	0.67
The UI/dashboard and controls for each experiment were responsive and intuitive	4.49	0.66
Overall Average Score	**4.34**	**0.73**

VI. CONCLUSION AND FUTURE WORK

This study presents the development of a remote lab system named RLabs that included the development of two use case experiments along with a software platform. The proposed system is qualitatively evaluated against seven NFAs - affordability, portability, scalability, compatibility, maintainability, usability, and universality. The experiments were built by retrofitting IoT components on traditional laboratory equipment. Modular and miniaturised versions of the same experiments are also built using 3D-printed components. Miniaturised experiments are lower in cost by 41 % and 24 % for Vanishing Rod and Focal Length experiments compared to the retrofitted setups, making them even more affordable. Similarly, there is also a reduction in weight and volume by 5.4 and 12 times for the Vanishing Rod experiment and 2.8 and 4.4 times for the Focal Length experiment, respectively, showing the compactness and portability of the miniaturised setups. The system is scalable as many experiments can be built at low-cost and with fewer materials. At the same time, the architecture of the software platform allows many experiments to be hosted without consuming many resources due to the implementation of the P2P live-streaming service. Moreover, the low latency of user experimentation and video streaming along with the lightweight server further enable RLabs to be scaled for more concurrent users easily. The compatibility of the system is shown by connecting different hardware boards (like Raspberry Pi, ESP32), IoT platforms (like Blynk and Thingspeak) and cameras (RaspiCam, USB camera and IP Camera). Also, the platform is tested by operating it on different devices, web browsers and operating systems. An automated testing suite has monitored the experiments' operational status for four months and reported an uptime of 84 %. The usability survey, filled out by a group of forty-five high school students, showed an average score of 4.34, indicating a positive learning experience and good usability of the system.

The domain of remote labs still poses research gaps and presents an extensive scope for future work. With the advent of natural language processing and large language models, it would be interesting to address the possibilities of providing a chat or dialogue-based user interface, allowing natural language conversations to enhance the learning experience of the users and allow multi-lingual support. Moreover, the provision of such a feature can also enable the aspect of enquiry-based learning by providing hints wherever required by the student to aid and guide the user in both performing the experiments and the appropriate analysis of the observed results. Such advancements could also look at real-time doubt clarification of the users while they are performing their experiments. The prospect of providing a voice-assisted interface for remote labs can also be explored. This also opens up opportunities to perform a thorough pedagogical analysis of the proposed RLabs system in the future.

ACKNOWLEDGMENT

The authors would like to thank Shikhar Educare, Amravati, Maharashtra, India, for providing valuable feedback on the RLabs system.

REFERENCES

[1] UNESCO. (2019). *Global Education Monitoring Report 2019: Migration, Displacement & Education: Building Bridges, Not Walls*. [Online]. Available: https://www.unesco.org/gem-report/en/migration

[2] World Bank. (2018). *World Development Report 2018: LEARNING To Realize Education's Promise*. [Online]. Available: https://www.worldbank.org/en/publication/wdr2018

[3] World Bank. (2020). *The COVID-19 Pandemic: Shocks To Education and Policy Responses*. [Online]. Available: https://www.worldbank.org/en/topic/education/publication/the-covid19-pandemic-shocks-to-education-and-policy-responses

[4] I. Angulo, L. Rodrìguez-Gil, and J. Garcìa-Zubìa, "Scaling up the lab: An adaptable and scalable architecture for embedded systems remote labs," *IEEE Access*, vol. 6, pp. 16887–16900, 2018.

[5] V.-M. Aitor, J. García-Zubía, I. Angulo, and L. Rodríguez-Gil, "Toward widespread remote laboratories: Evaluating the effectiveness of a replication-based architecture for real-world multiinstitutional usage," *IEEE Access*, vol. 10, pp. 86298–86317, 2022.

[6] L. Rodríguez-Gil, J. García-Zubia, P. Orduña, and D. Lopez-de-Ipiña, "An open and scalable web-based interactive live-streaming architecture: The WILSP platform," *IEEE Access*, vol. 5, pp. 9842–9856, 2017.

[7] F. Lustig, J. Dvorak, P. Kuriscak, and P. Brom, "Open modular hardware and software kit for creations of remote experiments accessible from PC and mobile devices," *Int. J. Online Biomed. Eng. (iJOE)*, vol. 12, no. 7, pp. 30–36, Jul. 2016. [Online]. Available: https://online-journals.org/index.php/i-joe/article/view/5833

[8] A. Villar-Martínez, L. Rodríguez-Gil, I. Angulo, P. Orduña, J. García-Zubía, and D. López-De-Ipiña, "Improving the scalability and replicability of embedded systems remote laboratories through a cost-effective architecture," *IEEE Access*, vol. 7, pp. 164164–164185, 2019.

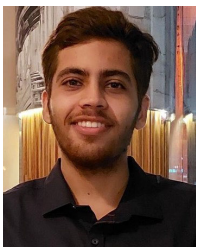
[9] C. Lavayssière, B. Larroque, and F. Luthon, "Laborem box: A scalable and open source platform to design remote lab experiments in electronics," *HardwareX*, vol. 11, Apr. 2022, Art. no. e00301. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2468067222000463

[10] J. B. D. Silva, G. D. Oliveira, I. N. D. Silva, P. M. Mafra, and S. M. S. Bilessimo, "Block.Ino: Remote lab for programming teaching and learning," *Int. J. Adv. Eng. Res. Sci.*, vol. 7, no. 1, pp. 41–47, 2020.

[11] J. Á. Ariza and S. G. Gil, "RaspyLab: A low-cost remote laboratory to learn programming and physical computing through Python and raspberry pi," *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, vol. 17, no. 2, pp. 140–149, May 2022.

[12] J. Álvarez Ariza and C. Nomesqui Galvis, "RaspyControl lab: A fully open-source and real-time remote laboratory for education in automatic control systems using raspberry PI and Python," *HardwareX*, vol. 13, Mar. 2023, Art. no. e00396. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2468067223000032
[13] J. Sáenz, J. Chacón, L. De La Torre, A. Visioli, and S. Dormido, "Open and low-cost virtual and remote labs on control engineering," *IEEE Access*, vol. 3, pp. 805–814, 2015.
[14] D. Reid, J. Burridge, D. Lowe, and T. Drysdale, "Open-source remote laboratory experiments for controls engineering education," *Int. J. Mech. Eng. Educ.*, vol. 50, no. 4, pp. 828–848, Oct. 2022, doi: 10.1177/03064190221081451.
[15] A. Van den Beemt, S. Groothuijsen, L. Ozkan, and W. Hendrix, "Remote labs in higher engineering education: Engaging students with active learning pedagogy," *J. Comput. Higher Educ.*, vol. 35, no. 2, pp. 320–340, Aug. 2023, doi: 10.1007/s12528-022-09331-4.
[16] A. Villar-Martínez, J. García-Zubía, I. Angulo, and L. Rodríguez-Gil, "Towards reliable remote laboratory experiences: A model for maximizing availability through fault-detection and replication," *IEEE Access*, vol. 9, pp. 45032–45054, 2021.
[17] S. Dormido, J. Sánchez, L. De la Torre, R. Heradio, C. Carreras, J. P. Sánchez, and M. Yuste, "Physics experiments at the UNEDLabs portal," *Int. J. Online Biomed. Eng. (iJOE)*, vol. 8, no. 1, pp. 26–27, Jan. 2012. [Online]. Available: https://online-journals.org/index.php/i-joe/article/view/1945
[18] L. de la Torre, J. Sánchez, S. Dormido, J. P. Sánchez, M. Yuste, and C. Carreras, "Two web-based laboratories of the FisL@bs network: Hooke's and Snell's laws," *Eur. J. Phys.*, vol. 32, no. 2, pp. 571–584, Feb. 2011, doi: 10.1088/0143-0807/32/2/027.
[19] H. Vargas, J. Sánchez, C. A. Jara, F. A. Candelas, F. Torres, and S. Dormido, "A network of automatic control web-based laboratories," *IEEE Trans. Learn. Technol.*, vol. 4, no. 3, pp. 197–208, Jul. 2011.
[20] A. Das, K. S. Viswanadh, R. Agrawal, A. Gureja, N. Nilesh, and S. Chaudhari, "Using miniature setups and partial streams for scalable remote labs," in *Proc. 10th Int. Conf. Future Internet Things Cloud (FiCloud)*, Aug. 2023, pp. 256–263.
[21] A. Gureja, R. Agrawal, S. Chaudhari, K. Vaidhyanathan, and V. Choppella, "Software architecture for multi-user multiplexing to enhance scalability in IoT-based remote labs," in *Proc. IEEE 9th World Forum Internet Things (WF-IoT)*, Oct. 2023, pp. 1–7.
[22] A. Shoufan, "Active distance learning of embedded systems," *IEEE Access*, vol. 9, pp. 41104–41122, 2021.
[23] M. Vallarino, S. Iacono, E. Bellanti, and G. V. Vercelli, "A flipped remote lab: Using a peer-assessment tool for learning 3-D modeling," *IEEE Trans. Learn. Technol.*, vol. 17, pp. 1140–1154, 2024.
[24] M. Schnieder, S. Ghosh, and S. Williams, "Using gamification and flipped classroom for remote/virtual labs for engineering students," in *Proc. 20th Eur. Conf. e-Learn.*, Berlin, Germany, 2021, pp. 28–29.
[25] V. Hayashi, J. Dutra, F. Almeida, R. Arakaki, E. Midorikawa, S. Canovas, P. Cugnasca, and W. Ruggiero, "Implementation of PjBL with remote lab enhances the professional skills of engineering students," *IEEE Trans. Educ.*, vol. 66, no. 4, pp. 369–378, Aug. 2023.
[26] Z. Lassoued, M. Alhendawi, and R. Bashitialshaaer, "An exploratory study of the obstacles for achieving quality in distance learning during the COVID-19 pandemic," *Educ. Sci.*, vol. 10, no. 9, p. 232, Sep. 2020. [Online]. Available: https://www.mdpi.com/2227-7102/10/9/232
[27] A. Satpathy. (2023). *What Are Non-functional Requirements and How To Build Them*. Accessed: 2023-12-05. [Online]. Available: https://www.modernrequirements.com/blogs/sub-topic/what-are-non-functional-requirements-and-how-to-build-them
[28] A. Marri, V. Nipane, R. Agrawal, S. Chaudhari, and P. Bhimalapuram, "Low-cost retrofitted IoT based titration setup for remote labs," in *Proc. 11th Int. Conf. Future Internet Things Cloud (FiCloud)*, Aug. 2024, pp. 109–116.
[29] S. J. Ling, J. Sanny, and W. Moebs, *University Physics*, vol. 3. Houston, TX, USA: OpenStax, Sep. 2016. [Online]. https://openstax.org/books/university-physics-volume-3/pages/1-introduction
[30] P. Kumar. (2021). *MIPI Camera Vs USB Camera—A Detailed Comparison*. [Online]. Available: https://www.e-consystems.com/blog/camera/technology/mipi-camera-vs-usb-camera-a-detailed-comparison/
[31] M. Scarpino, *Motors for Makers: A Guide To Steppers, Servos, and Other Electrical Machines*. Indianapolis, IN, USA: QUE, 2015. [Online]. Available: https://ptgmedia.pearsoncmg.com/images/9780134032832/samplepages/9780134032832.pdf
[32] B. Shankar, M. K. Sarithlal, S. Sharat, J. Freeman, and K. Achuthan, "Remote triggered virtual laboratory for Hooke's law using LabVIEW," in *Proc. 39th Annu. Conf. IEEE Ind. Electron. Soc.*, Nov. 2013, pp. 3729–3734.
[33] M. Kalúz, L. Čirka, R. Valo, and M. Fikar, "ArPi lab: A low-cost remote laboratory for control education," *IFAC Proc. Volumes*, vol. 47, no. 3, pp. 9057–9062, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474667016430430
[34] J. M. Sierra-Fernández, O. Florencias-Oliveros, M. J. Espinosa-Gavira, J. C. Palomares-Salas, A. Agüera-Pérez, and J. J. González-de-la-Rosa, "Reconfigurable web-interface remote lab for instrumentation and electronic learning," in *Proc. IEEE Global Eng. Educ. Conf. (EDUCON)*, Apr. 2020, pp. 713–717.
[35] K. S. Viswanadh, O. Kathalkar, P. Vinzey, N. Nilesh, S. Chaudhari, and V. Choppella, "CV and IoT-based remote triggered labs: Use case of conservation of mechanical energy," in *Proc. 9th Int. Conf. Future Internet Things Cloud (FiCloud)*, Aug. 2022, pp. 100–106.
[36] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.
[37] Sepfy. (2021). *GitHub—Sepfy/Libpeer: WebRTC Library for IoT/Embedded Device Using C*. [Online]. Available: https://github.com/sepfy/libpeer/tree/main
[38] T. P. Hut. (2021). *Raspberry Pi Won't Connect To WiFi*. [Online]. Available: https://support.thepihut.com/hc/en-us/articles/360014695877-Raspberry-Pi-won-t-connect-to-WiFi
[39] A. Chevalier, C. Copot, C. Ionescu, and R. De Keyser, "A three-year feedback study of a remote laboratory used in control engineering studies," *IEEE Trans. Educ.*, vol. 60, no. 2, pp. 127–133, May 2017.
[40] J. Cuadros, V. Serrano, J. García-Zubía, and U. Hernandez-Jayo, "Design and evaluation of a user experience questionnaire for remote labs," *IEEE Access*, vol. 9, pp. 50222–50230, 2021.

SAVITHA VISWANADH KANDALA received the bachelor's and Master of Science (by Research) degrees in electronics and communication engineering from the International Institute of Information Technology Hyderabad (IIIT-H), India. He is currently pursuing the Ph.D. degree in computer science with the School of Computing, National University of Singapore (NUS). His research interests include developing foundational models for wireless and embedded systems. He received the TIH-IoT Chanakya Fellowship Award, in 2023.



AKSHIT GUREJA is currently pursuing the bachelor's and Master of Science (by Research) degrees in electronics and communication engineering from the International Institute of Information Technology Hyderabad, India. His current research interests include software platforms for Internet of Things and its applications. He received the TIH-IoT Chanakya Fellowship Award, in 2023.

NAGESH WALCHATWAR received the bachelor's degree in electronics and telecommunication engineering from the St. Vincent Pallotti College of Engineering and Technology, Nagpur. He is currently pursuing the Master of Science (by Research) degree with the International Institute of Information Technology Hyderabad, India. His current research interests include automated CI/CD pipelines and security for Internet of Things (IoT).

RISHABH AGRAWAL received the Bachelor of Technology degree in electronics and communication from the International Institute of Information Technology Hyderabad, in 2024. His research interests include the Internet of Things and designing modular and sustainable IoT devices. He received the TIH-IoT Chanakya Fellowship Award from the TIH Foundation for IoT and IoE, in 2023.

SHIVEN SINHA is currently pursuing the B.Tech. and M.S. (by Research) degrees in computer science and engineering from the International Institute of Information Technology Hyderabad, India. His research interests include machine learning and its applications in mathematics and science.

SACHIN CHAUDHARI (Senior Member, IEEE) received the B.E. degree in electronics from Visvesvaraya National Institute of Technology (VNIT), Nagpur, India, in 2002, the M.E. degree in telecommunication from Indian Institute of Science (IISc), Bangalore, India, in 2004, and the D.Sc. (Tech.) degree from Aalto University (formerly TKK), Finland, in 2012. Between August 2004 and May 2007, he was with Esqube Communications, Bangalore, India, as a Senior Wireless Communication Engineer. During 2013 to 2014, he was a Post-Doctoral Researcher at Aalto University. He joined the International Institute of Information Technology (IIIT), Hyderabad, India, in December 2014, where he is currently an Associate Professor. His research interests are in the field of signal processing and machine learning for wireless communication and particularly, in the physical layer aspects of the Internet of Things (IoT), 5G/6G, satellite communications, and cognitive radio. During 2019 to 2022, he was the coordinator of the Center of Excellence on IoT for Smart Cities at IIITH, which was supported by the India-EU collaboration project, ETSI, and TSDSI. Currently, he is the coordinator of the 5G Use Case Lab setup at IIIT-H by the Department of Telecommunication (DoT). He is also actively involved in India's First Living Laboratory for Smart City Research at IIITH.

KARTHIK VAIDHYANATHAN is currently an Assistant Professor with the Software Engineering Research Center, IIIT-Hyderabad, India, where he is also associated with the leadership team of smart city living laboratory. His main research interests include the intersection of software architecture and machine learning, with a specific focus on building sustainable software systems, how machine learning techniques can be leveraged to better architect self-adaptive systems, and further how to better define architecting practices for developing machine learning-enabled software systems. As a part of his research activities, he serves as a reviewer/organizing committee member in various workshops, conferences, and journals. He is also an Editorial Board Member (SE Radio) of IEEE Software. He also poses more than five years of industrial experience as an Employee and as a Consultant in building and deploying ML products/services.

VENKATESH CHOPPELLA received the bachelor's degree from Indian Institute of Technology Kanpur, the master's degree from Indian Institute of Technology Madras, and the Ph.D. degree in computer science from Indiana University, Bloomington, USA. He is currently an Associate Professor with the Software Engineering Research Center, IIIT Hyderabad, India. His current research interests include formal methods and software architectures and the design and implementation of virtual laboratories.

PRABHAKAR BHIMALAPURAM received the Ph.D. degree from Cornell University, focusing on thermodynamics, statistical mechanics, and computational sciences. He is currently an Assistant Professor with the Center for Computational Natural Sciences and Bioinformatics, IIIT Hyderabad. His research interests include molecular dynamics and thermodynamic modeling, contributing significantly to understanding complex physical and biological phenomena. He is known for his interdisciplinary research and active involvement in mentoring and academic activities with IIIT Hyderabad.

HARIKUMAR KANDATH (Member, IEEE) received the Ph.D. degree in aerospace engineering from Indian Institute of Science (IISc), Bengaluru, India, in 2015. He is currently an Assistant Professor with the Robotics Research Centre, IIIT Hyderabad, India. His research interests include the applications of control theory to unmanned systems and flight dynamics.

AFTAB HUSSAIN (Member, IEEE) received the bachelor's degree from IIT Roorkee, in 2009, and the M.S. and Ph.D. degrees from KAUST, in 2016.

He is currently an Assistant Professor with the International Institute of Information Technology (IIIT) Hyderabad and the Principal Investigator of the PATRIoT Laboratory. He has more than nine years of experience working with device fabrication, thin film characterization, and device testing, along with sensor applications in IoT and related areas. His research interests include flexible and stretchable electronics and their applications in various IoT domains, such as healthcare, mobility, and smart cities. He has more than 80 scientific publications, holds four granted U.S. patents, along with six others in various stages of filing. He is a member of the Executive Committee (ExeCom) of IEEE CAS/EDS Joint Chapter in Hyderabad and an Associate Editor (Flexible Electronics) of *Frontiers in Electronics* journal.